



AFRL-RI-RS-TR-2016-004

COMPUTING ON ENCRYPTED DATA: THEORY AND APPLICATION

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

JANUARY 2016

FINAL TECHNICAL REPORT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

STINFO COPY

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE**

NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nationals. Copies may be obtained from the Defense Technical Information Center (DTIC) (<http://www.dtic.mil>).

AFRL-RI-RS-TR-2016-004 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE DIRECTOR:

/ S /

CARL THOMAS
Work Unit Manager

/ S /

MARK LINDERMAN
Technical Advisor, Computing &
Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
<small>The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.</small>					
1. REPORT DATE (DD-MM-YYYY) JANUARY 2016		2. REPORT TYPE FINAL TECHNICAL REPORT		3. DATES COVERED (From - To) MAY 2011 – JUN 2015	
4. TITLE AND SUBTITLE COMPUTING ON ENCRYPTED DATA: THEORY AND APPLICATION				5a. CONTRACT NUMBER FA8750-11-2-0225	
				5b. GRANT NUMBER N/A	
				5c. PROGRAM ELEMENT NUMBER 62303E	
6. AUTHOR(S) Shafi Goldwasser and Vinod Vaikuntanathan				5d. PROJECT NUMBER BL11	
				5e. TASK NUMBER OM	
				5f. WORK UNIT NUMBER IT	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Massachusetts Institute of Technology 77 Massachusetts Ave Cambridge, MA 02139-4301				8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory/RITA 525 Brooks Road Rome NY 13441-4505				10. SPONSOR/MONITOR'S ACRONYM(S) AFRL/RI	
				11. SPONSOR/MONITOR'S REPORT NUMBER AFRL-RI-RS-TR-2016-004	
12. DISTRIBUTION AVAILABILITY STATEMENT Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This report documents the development of second generation Fully Homomorphic Encryption (FHE) schemes providing simple and efficient FHE operations based on standard cryptographic hardness assumptions. These solutions form the current state-of-the-art in fully homomorphic encryption. This was further developed into attribute based encryption schemes, reusable garbled circuits, functional encryption schemes and secure multi-party computations solutions.					
15. SUBJECT TERMS Fully Homomorphic Encryption (FHE), Secure Multiparty Computation (SMC), Learning with Errors (LWE), Attribute Based Encryption (ABE), Predicate Encryption (PE), Functional Encryption (FE), Pseudorandom Functions (PRFs)					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UU	18. NUMBER OF PAGES 662	19a. NAME OF RESPONSIBLE PERSON CARL THOMAS
a. REPORT U	b. ABSTRACT U	c. THIS PAGE U			19b. TELEPHONE NUMBER (Include area code) NA

Table of Contents

1	Summary	1
2	Introduction	2
3	Methods, Assumptions and Procedures	3
4	Results and Discussion	5
4.1	Results: Fully Homomorphic Encryption	5
4.1.1	Second Generation FHE: Simpler, Faster, Stronger	7
4.1.2	Third Generation FHE: Best Possible Assumptions	8
4.1.3	Multi-key FHE and On-the-Fly Multiparty Computation	8
4.1.4	Practical HE: Machine Learning on Encrypted Data	9
4.2	Results: Functional Encryption	9
4.2.1	Attribute-based Encryption	10
4.2.2	Bounded-Key Functional Encryption	11
4.2.3	Succinct Functional Encryption and Reusable Garbled Circuits	11
4.2.4	Multi-Input Functional Encryption	12
4.3	Results: Large-Scale Multiparty Computation	12
4.4	Results: Leakage-Resilient Computation	13
4.4.1	Goldwasser-Rothblum Leakage-Resilience Compiler	14
4.4.2	Leakage-Resilient Multiparty Computation	14
4.5	Results: Functional Signatures and Pseudorandom Functions	15
4.5.1	Functional Signatures and Pseudorandom Functions	15
4.5.2	Constrained PRFs for Arbitrary Circuits from LWE	16
4.5.3	Aggregate Pseudo-random Functions and Connections to Learning Theory	16
5	Conclusions and Recommendations	17
6	Bibliography	18
7	Appendix - Papers	
7.1	Efficient Fully Homomorphic Encryption from (Standard) LWE	27
7.2	Fully Homomorphic Encryption without Bootstrapping	65
7.3	On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption	92
7.4	Lattice-Based FHE as Secure as PKE	165
7.5	Machine Learning Classification over Encrypted Data	186
7.6	Functional Encryption with Bounded Collusions via Multi-Party Computation	220
7.7	Attribute-Based Encryption for Circuits	258
7.8	Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE, and Compact Garbled Circuits	293
7.9	Reusable Garbled Circuits and Succinct Functional Encryption	334
7.10	Multi-Input Functional Encryption	385
7.11	Optimally Resilient and Adaptively Secure Multi-Party Computation with Low Communication Locality	426
7.12	Functional Signatures and Pseudorandom Functions	451

Table of Contents (Continued)

Appendix - Papers (Continued)

7.13 Aggregate Pseudorandom Functions and Connections to Learning	485
7.14 Constrained Key-Homomorphic PRFs from Standard Lattice Assumptions Or: How to Secretly Embed a Circuit in Your PRF	523
7.15 Communication Locality in Secure Multi-party Computation, How to Run Sublinear Algorithms in a Distributed Setting	552
7.16 How to Compute in the Presence of Leakage	573
7.17 Multiparty Computation Secure Against Continual Memory Leakage	636
8. Glossary of Terms	656

List of Figures

Figure 1: A list of major FHE schemes. All encryption schemes are *leveled FHE* schemes, namely they support evaluation of circuits of a-priori bounded depth. They can be generically converted into a *pure FHE* using Gentry’s bootstrapping method [Gen09], assuming that the underlying leveled FHE scheme is *circular secure*. The rows in gray represent contributions from our team. SSS denotes the sparse subset sum assumption, BDD the bounded distance decoding assumption, GCD is greatest common divisors, LWE is learning with errors and NTRU is the N-th order truncated ring encryption scheme. 6

Figure 2: Attribute-based, Predicate and Functional Encryption Schemes. The first six rows are ABE, the next two PE and the rest are FE schemes. The rows in gray represent contributions from our team. 11

Summary

This report describes the results we obtained as a result of our project “Computing on Encrypted Data: Theory and Applications” as part of the DARPA PROCEED program. Our results form a strong theoretical foundation of the science of computing on encrypted data.

A major outcome of our project was the invention of the second generation FHE schemes which gave us several simple and efficient FHE schemes based on standard cryptographic hardness assumptions. Our solutions [BV11a, BGV12, LTV12] form the state of the art in fully homomorphic encryption, and formed the backbone of homomorphic encryption implementations in the PROCEED program. In addition, they have also been implemented as part of the open source homomorphic encryption library HELib.

Going forward, we developed new cryptographic primitives that achieve the dual goals of supporting sophisticated functionalities while admitting efficient and practical constructions, including the first general-purpose attribute-based encryption [GVW13], reusable garbled circuits [GKP⁺13] and several types of functional encryption [GVW12, GVW15] schemes. Our project also contributed to the development of novel secure multi-party computation solutions.

Introduction

We live in a world where information and computation is at our fingertips, but also a world where our personal data is stored and processed in highly adversarial environments. While we now have cryptographic methods such as public-key encryption, digital signatures and secure protocols that form the basis of secure electronic transactions, traditional cryptography is inadequate to handle the challenges posed by modern technologies. In particular, the emerging paradigm of *cloud computing* lets us outsource storage and computation to powerful third-party servers, but raises serious privacy concerns. Of course, encrypting the data we hand over to the cloud protects its privacy, but how then can the cloud compute on the encrypted data?

The answer lies in *fully homomorphic encryption* (FHE), a special type of encryption system where one can perform arbitrarily complex computations on encrypted data without ever decrypting it. Long considered the unattainable holy grail of cryptography, this primitive was recently realized in the ground-breaking work of Craig Gentry in 2009. Unfortunately, Gentry's construction suffers from inherent limitations in efficiency, was considered impractical and raised widespread speculation that computing on encrypted data might never see the light of day.

A major outcome of our project has been the invention of the **second generation** FHE schemes which gave us several **simple and efficient** FHE schemes **based on standard cryptographic hardness assumptions**. Our solutions [BV11a, BGV12, LTV12] form the state of the art in fully homomorphic encryption, and formed the backbone of homomorphic encryption implementations in the PROCEED program. In addition, they have also been implemented as part of the open source homomorphic encryption library HELib.

Going forward, we developed new cryptographic primitives that achieve the dual goals of supporting *sophisticated functionalities* while admitting *efficient and practical* constructions. While fully homomorphic encryption permits *any* computation on encrypted data, we would like to provide *fine-grained control* over what types of functions can be computed. This ability comes in handy in a number of scenarios, such as ensuring the correctness of outsourced computations. We constructed the first general-purpose **attribute-based encryption** [GVW13], **reusable garbled circuits** [GKP⁺13] and several types of **functional encryption** [GVW12, GVW15] schemes that address these issues.

Our project also contributed to the development of novel secure multi-party computation solutions. Multiparty computation is a notion that was introduced and studied in cryptography from the 1980s [Yao86, GMW87, BGW88], and provides a generally more efficient, but interactive, alternative to the problem of computing on encrypted data. We developed several communication-local MPC protocols that handle a very large number of users, and in addition, withstand leakage of the internal states of 99% of the participants.

Our results form a strong theoretical foundation of the science of computing on encrypted data, powerful enough to address the new and emerging challenges. We now proceed to describe our results in detail.

Methods, Assumptions and Procedures

Our methods are largely algorithmic. Meaning, we construct algorithms (sometimes interactive algorithms, which we call protocols) that solve several cryptographic problems and provide formal security proofs under well-defined and well-studied computational hardness assumptions. We now state our assumptions, together with some mathematical preliminaries below.

Lattices and lattice problems. A lattice is a discrete additive subgroup of \mathbb{R}^n . A lattice can be characterized as the *integer span of basis vectors*, usually denoted as a matrix $\mathbf{B} \in \mathbb{R}^{n \times \ell}$ (ℓ is the *rank* of the lattice, for simplicity we assume that $\ell = n$, but all definitions carry over to the general case). The minimum distance between two lattice points is equal to the length of the shortest nonzero lattice point and is denoted by $\lambda_1 = \lambda_1(\mathbf{B})$ (many norms can be considered, most commonly ℓ_2). The shortest vector problem $\text{SVP}_\gamma(\mathbf{B})$ is to find a lattice vector of length at most $\gamma \cdot \lambda_1$ given \mathbf{B} . The decision version $\text{GapSVP}_\gamma(\mathbf{B}, d)$ is the promise problem which accepts if $\lambda_1 \leq d$ and rejects if $\lambda_1 > \gamma \cdot d$. The closest vector problem $\text{CVP}_\gamma(\mathbf{B}, \mathbf{t})$ is to find the closest lattice vector to the target \mathbf{t} , up to an approximation factor γ . The decision version $\text{GapCVP}_\gamma(\mathbf{B}, \mathbf{t}, d)$ is defined analogously. The parameter $\gamma = \gamma(n)$ is called the approximation factor, and controls the hardness of GapSVP_γ (increasing γ makes it easier). It was known early on that when $\gamma = \mathcal{O}(1)$, GapSVP_γ is NP-complete, while when $\gamma = 2^n$, it is solvable in polynomial time. Ajtai's original work on lattice cryptography gave a one-way function based on the hardness of GapSVP_γ when $\gamma = \text{poly}(n)$.

An *ideal lattice* is a lattice that is defined by an *embedding* into \mathbb{R}^n of an ideal in the ring of integers of some number field.

Learning with Errors (LWE). The problem is defined with respect to parameters q, n, χ (some variants require additional parameters). Let \mathbb{Z}_q be the ring of integers modulo q and let χ be a distribution over \mathbb{Z}_q (referred to as the *noise distribution*). The parameter $n \in \mathbb{N}$ is the *dimension* of the problem. For all $\mathbf{s} \in \mathbb{Z}^n$ consider the oracle $A_{\mathbf{s}, \chi}$ that for every call returns (\mathbf{a}, b) where $\mathbf{a} \in \mathbb{Z}_q^n$ is uniformly distributed and $b = \langle \mathbf{a}, \mathbf{s} \rangle + e \pmod{q}$, where e is drawn from χ .

The search LWE problem $\text{LWE}_{n,q,\chi}$ is to retrieve a uniformly distributed \mathbf{s} given oracle access to $A_{\mathbf{s}, \chi}$. The decision LWE problem $\text{DLWE}_{n,q,\chi}$ is to distinguish between oracle access to $A_{\mathbf{s}, \chi}$ (for a uniformly distributed \mathbf{s}) and oracle access to a random oracle \mathcal{O} that on every call returns a uniform element in $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

The hardness of solving search LWE with *discrete gaussian* noise distribution has been shown to be related to the hardness of GapSVP_γ via quantum [Reg05] and classical [Pei09] reductions. The classical reduction results in a worse parameter γ and as is, it is only applicable to $q \approx 2^n$. Search to decision connections are known for many values of q .

Ring Learning with Errors (Ring LWE). The ring learning with errors (RLWE) problem was introduced by Lyubashevsky, Peikert and Regev. Here, let R denote the ring of integers of a number field, namely $R = \mathbb{Z}[x]/f(x)$ for some cyclotomic $f(x)$. For all $s \in R_q := R/qR$ consider the oracle $A_{s,\chi}$ that for every call returns (\mathbf{a}, \mathbf{b}) where $\mathbf{a} \in R_q$ is uniformly distributed and $\mathbf{b} = \mathbf{a} \cdot s + \mathbf{e}$ (over R_q), where \mathbf{e} is drawn from χ (here, χ is a distribution over the ring of integers of the number field R).

The search Ring LWE problem $\text{RLWE}_{n,q,\chi}$ is to retrieve a uniformly distributed s given oracle access to $A_{s,\chi}$. The decision Ring LWE problem $\text{DLWE}_{n,q,\chi}$ is to distinguish between oracle access to $A_{s,\chi}$ (for a uniformly distributed s) and oracle access to a random oracle \mathcal{O} that on every call returns a uniform element in $\mathbb{Z}_q^n \times \mathbb{Z}_q$.

Results and Discussion

4.1 Results: Fully Homomorphic Encryption

The evolution of fully homomorphic encryption schemes can be roughly partitioned into three *generations*.

First Generation. The first generation of fully homomorphic encryption (FHE) started with the breakthrough work of Gentry [Gen09] and was followed by several simplifications [vDGHV10, SV09, BV11b]. This sequence of works had two major disadvantages. First, they had extremely large key-sizes and extremely slow algorithms. Secondly, the security of these schemes were based on complex, little-studied assumptions. We believe that these two issues are correlated: indeed, the schemes relied on complex hardness assumptions because we lacked a fundamental understanding of the basis for their security. In turn, this caused us to add on “band-aids” that had negative effects on efficiency.

Second Generation. A major contribution of our project was the invention of the second generation of fully homomorphic encryption, in a sequence of papers [BV11a, BGV12] which we describe in detail below. These works not only constructed (leveled) FHE schemes from standard computational assumptions, but also offered orders of magnitude better efficiency. These schemes were quickly built upon, most notably in a sequence of works by Gentry, Halevi and Smart [GHS12a, GHS12b] who showed powerful techniques for “SIMD” evaluation, namely methods to pack many inputs into one ciphertext and to homomorphically evaluate on them in parallel.

A leveled FHE is one which can evaluate circuit of a-priori bounded polynomial depth. One can convert any leveled FHE scheme into a pure FHE scheme that can evaluate arbitrary circuits, using Gentry’s bootstrapping theorem [Gen09]. This involves making an additional hardness assumption, namely the “circular security” of the leveled HE scheme. We refer the reader to [BV11a] for more details.

Third Generation. Finally, a third generation of (leveled) FHE schemes emerged with the work of Gentry, Sahai and Waters [GSW13]. Even simpler to describe than the second generation FHE schemes, these were extended in our work [BV14] to result in FHE schemes that relied on the worst-case hardness of approximating shortest vectors to within polynomial factors, or equivalently, the hardness of LWE with a polynomial modulus. These schemes enjoy a very slow noise growth, and hence result in better parameters. There have since been several extensions of these schemes [AP14, DM15].

Scheme	Assumption	Note
[Gen09]	av-case ideal-BDD and SSS	
[SV09]	av-case principal-ideal-BDD and SSS	Improved efficiency of [Gen09]
[vDGHV10]	av-case approximate GCD and SSS	simple to describe
[BV11b]	Ring-LWE and SSS	
[BV11a]	(sub-exponential) LWE	efficient instantiation from Ring-LWE introduced dimension/modulus switching
[GH11]	av-case ideal-BDD	
[BGV12]	(super-polynomial) LWE	efficient instantiation from Ring-LWE
[Bra12]	(super-polynomial) LWE	efficient instantiation from Ring-LWE
[LTV12]	NTRU assumption	<i>also multi-key HE</i>
[GHS12a, GHS12b]	Ring LWE	SIMD and polylog overhead HE
[BLLN13]	NTRU = Ring-LWE	based [LTV12] on Ring LWE
[GSW13]	(super-polynomial) LWE	efficient instantiation from Ring-LWE
[BV14]	(polynomial) LWE	efficient instantiation from Ring-LWE
[AP14]	(polynomial) LWE	efficient instantiation from Ring-LWE

Figure 1: A list of major FHE schemes. All encryption schemes are *leveled FHE* schemes, namely they support evaluation of circuits of a-priori bounded depth. They can be generically converted into a *pure FHE* using Gentry’s bootstrapping method [Gen09], assuming that the underlying leveled FHE scheme is *circular secure*. The rows in gray represent contributions from our team. SSS denotes the sparse subset sum assumption, BDD the bounded distance decoding assumption, GCD is greatest common divisors, LWE is learning with errors and NTRU is the N-th order truncated ring encryption scheme.

4.1.1 Second Generation FHE: Simpler, Faster, Stronger

We describe here the sequence of works [BV11a, BGV12]. The first of these works with Brakerski and Vaikuntanathan [BV11a], for the first time, showed how to construct a leveled homomorphic encryption scheme solely under the learning with errors (LWE) assumption. All previous constructions were natively able to evaluate circuits of depth $O(\log n)$ where n is the security parameter, and extending these into fully homomorphic schemes required making additional assumptions.

The starting point of our construction is Regev’s encryption scheme in which the ciphertext encrypting a bit $m \in \{0, 1\}$ is of the form

$$(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e + m \lfloor q/2 \rfloor)$$

where $\mathbf{a} \in \mathbb{Z}_q^n$ is a uniformly random vector, $\mathbf{s} \in \mathbb{Z}_q^n$ is the secret key, and $e \leftarrow \chi$ is a “small” error term drawn from an error distribution χ (typically a discrete Gaussian distribution). All computations here are performed mod q . It is not hard to see that this scheme is, by itself, additively homomorphic.¹ Our main contribution is to show how to do homomorphic multiplication. In particular, our HE construction uses *two new ideas*, both of which have found uses in later designs of homomorphic encryption and elsewhere.

Relinearization (or Dimension Reduction). We first observe that a tensor product of two ciphertexts is an encryption of the product of two messages, albeit under a longer key, namely $\mathbf{s} \otimes \mathbf{s}$, the tensor product of the secret key with itself. However, in doing so, the size of the ciphertext increases dramatically, to $O(n^2)$ from $O(n)$. Clearly, this is not sustainable and thus, we come up with a way to reduce the dimension back to n .

Roughly speaking, this is done by publishing several “hints” in the public key which consist of encryptions of the quadratic monomials in the coefficients of \mathbf{s} using a different secret key \mathbf{s}' . Of course, the key idea is in doing so without compromising the security of the overall system at all.

Modulus Switching (or Modulus Reduction). Relinearization by itself only gives us a somewhat homomorphic encryption scheme capable of evaluating $O(\log n)$ depth circuits since the ciphertext error squares every level of homomorphic multiplication. With d levels, the noise becomes $B_0^{2^d}$ where B_0 is the magnitude of the initial ciphertext noise. Now,

$$B_0^{2^d} < q/4 < 2^{n^\epsilon} \quad (4.1)$$

for some constant $0 < \epsilon < 1$. Here, the first inequality is to ensure correctness of decryption, and the second to ensure security. This gives us $d \approx \epsilon \log n$. Unfortunately, this turns out to be not quite enough to apply bootstrapping [Gen09] and obtain an FHE scheme.

The second major idea in [BV11a] is a way to slow down the noise growth during homomorphic multiplication. The basic idea is simple and two-fold. First, the modulus does not really matter in LWE as one can always scale the LWE samples down to the “torus” $[0, 1)$. This observation has been made in several prior works including [Reg05]. Second, multiplying two error terms that live in $[0, 1)$ only decreases their magnitude, as opposed to increasing! This surprisingly simple observation gives us a noise growth of $B_0 \rightarrow B_0 \cdot \text{poly}(n)$ as opposed to $B_0 \rightarrow B_0^2$.

Equation 4.1 now becomes

$$B_0 \cdot n^{O(d)} < q/4 < 2^{n^\epsilon} \quad (4.2)$$

¹The reader might notice that when adding two ciphertexts, the error terms increase in magnitude. Indeed, this is an issue which limits the total number of homomorphic operations we can perform on the ciphertexts. However, setting q to be slightly super-polynomial and the magnitude of errors from χ to be polynomial, allows us to do a superpolynomial number of additions.

resulting in $d \approx n^\epsilon / \log n$.

Overall, the result is the first leveled FHE scheme that assumes only the hardness of learning with errors (LWE). With bootstrapping, and using the fact that the decryption circuit can be written as a Boolean circuit of depth $d = O(\log n)$, the resulting scheme relies on the hardness of $\text{lwe}_{n,q,\chi}$ where $q = n^{O(\log n)}$ and χ is any distribution with $\text{poly}(n)$ -bounded support.

The ideas underlying the [BV11a, BGV12] scheme can be ported just as well to other settings, such as ring LWE, NTRU and approximate GCD resulting in several variants. (Indeed, the ring LWE variant was described explicitly in [BGV12]). In practice, one should use either the ring LWE or the NTRU variant for reasons of efficiency.

4.1.2 Third Generation FHE: Best Possible Assumptions

All FHE schemes prior to the third generation relied on hardness assumptions that are *quantitatively worse* than those needed for public key encryption. In particular, the scheme of [BGV12] described above relies on the hardness of approximating lattice problems to within $n^{O(\log n)}$ factor.

Building on an FHE scheme of Gentry, Sahai and Waters [GSW13], Brakerski and PI Vaikuntanathan show that (leveled) FHE can be based on the hardness of $O(n^{2+\epsilon})$ -approximating lattice problems such as GapSVP (under classical reductions). This matches the best known hardness for regular (non-homomorphic) lattice based public-key encryption up to the ϵ factor. As usual, a circular security assumption can be used to achieve a non-leveled (pure) FHE scheme.

Our approach consists of three main ideas: Noise-bounded sequential evaluation of high fan-in operations; Circuit sequentialization using Barrington's Theorem; and finally, successive dimension-modulus reduction. In particular, the first and the most important of these ideas results in a dramatic slow-down in noise growth during homomorphic multiplication. Whereas before, multiplying N ciphertexts with a noise magnitude of B increased the error to $O(B^{\log N})$ (using a binary multiplication tree), it turns out that in our scheme, the error only increases to $O(BNn)$ where n is the security parameter. Surprisingly, this is nearly as small as the error increase during homomorphic addition!

Our scheme has since been improved on, first to remove the use of Barrington's theorem and the associated inefficiency [AP14] and secondly to implement bootstrapping in under a second [DM15].

4.1.3 Multi-key FHE and On-the-Fly Multiparty Computation

Homomorphic encryption enables an untrusted server to compute on ciphertexts encrypted under a single user's key. Often in practice, we want to compute on data belonging to multiple users who encrypt it under their own keys. Indeed, the most valuable computations are of this nature – hospitals who want to collaborate on their private datasets on rare diseases, financial institutions that want to collaborate and discover global trends in the market, and so on.

We define a new type of encryption scheme that we call multikey FHE, which is capable of operating on inputs encrypted under multiple, unrelated keys. A ciphertext resulting from a multikey evaluation can be jointly decrypted using the secret keys of all the users involved in the computation. We construct a multikey FHE scheme based on NTRU [HPS98], a very efficient public-key encryption scheme proposed in the 1990s. It was previously not known how to make NTRU fully homomorphic even for a single party. *Indeed, our fully homomorphic system based on NTRU is the leading candidate for a practical FHE scheme* as has been demonstrated in several followup works [BLLN13, DDS14, DÖSS15], some of which are part of the PROCEED program [RC14].

We also define a new notion of secure multiparty computation aided by a computationally-powerful, but untrusted "cloud" server. In this notion that we call *on-the-fly multiparty computation* (MPC), the cloud can non-interactively perform arbitrary, dynamically chosen computations on data belonging to arbitrary sets of users chosen on-the-fly. All user's input data and intermediate results are protected from snooping by the cloud as well as other users. This extends the standard notion of fully homomorphic encryption (FHE), where users can only enlist the cloud's help in evaluating functions on their own encrypted data.

In on-the-fly MPC, each user is involved only when initially uploading his (encrypted) data to the cloud, and in a final output decryption phase when outputs are revealed; the complexity of both is independent of the function being computed and the total number of users in the system. When users upload their data, they need not decide in advance which function will be computed, nor who they will compute with; they need only retroactively approve the eventually-chosen functions and on whose data the functions were evaluated.

This notion is qualitatively the best possible in minimizing interaction, since the users' interaction in the decryption stage is inevitable: we show that removing it would imply generic (virtual black-box) program obfuscation and is thus impossible.

While our construction was based on the NTRU assumption, there has since been a novel construction of multi-key FHE from LWE and Ring LWE assumptions [CM15, MW15].

4.1.4 Practical HE: Machine Learning on Encrypted Data

Machine learning classification is used in numerous settings nowadays, such as medical or genomics predictions, spam detection, face recognition, and financial predictions. Due to privacy concerns in some of these applications, it is important that both the data and the classifier remain confidential.

In work by PI Goldwasser and collaborators [BPTG15], we construct three major classification protocols that satisfy privacy constraints: hyperplane decision, Naïve Bayes, and decision trees. These protocols may also be combined with AdaBoost. They rely on a new library of building blocks for constructing classifiers securely. We demonstrate the versatility of this library by constructing a face detection classifier. In addition, our protocols are efficient, taking milliseconds to a few seconds to perform a classification when running on real medical datasets.

We remark that supervised learning algorithms consist of two phases: (i) the training phase during which the algorithm learns a model w from a data set of labeled examples, and (ii) the classification phase that runs a classifier C over a previously unseen feature vector x , using the model w to output a prediction $C(x, w)$. In applications that handle sensitive data, it is important that the feature vector x and the model w remain secret to one or some of the parties involved. Our protocols pertain to the second classification phase.

4.2 Results: Functional Encryption

Homomorphic encryption opens the door to many exciting applications, but also raises questions as to its ultimate usefulness. Perhaps the biggest such question is:

Who can decrypt the result of computations on encrypted data?

Although computation on encrypted data using FHE can be performed by anyone, only the holder of the secret key can decrypt the result of a computation; the secret key, however, allows decryption of the *entire* data and not just the result. This rules out a large class of applications in which the party computing on the encrypted data needs to determine the computation result on its own, but should not know anything else about the input, and should not assume that the secret key owner is online to help him decrypt. This leads to

the question: can we selectively reveal chosen functions of the data while keeping all other information hidden? This question truly exemplifies the essence of the paradigm of computing on encrypted data, namely, the delicate balance between keeping data private on the one hand, and revealing carefully chosen functions of it on the other hand.

A promising approach to this problem is functional encryption [SW05, GPSW06, KSW13, BSW12] where the holder of the secret key can generate and provide others with keys for functions, for example, sk_f for a function f . Anyone with access to the key sk_f and an encryption of x can obtain $f(x)$, but nothing more about x . In fact, we will require that an adversary who obtains polynomially many secret keys $sk_{f_1}, sk_{f_2}, \dots, sk_{f_n}$ and an encryption of x can learn $f_1(x), \dots, f_n(x)$ but nothing else. This is called *many key security* or *collusion resistance*, whereas the former requirement is called *single key security*.

There are many variants we will be interested in which differ in the functionality and security properties.

- **Attribute-based Encryption:** In an attribute-based encryption (ABE) scheme [SW05, GPSW06], one encrypts a payload data M relative to a set of attributes x . Given the secret key for a Boolean function (also called a predicate) f , one can decrypt and obtain the payload M if and only if $f(x) = 1$. Attribute-based encryption *does not require that the encryption hides the attributes*, only that it hides the payload.
- **Predicate Encryption:** A predicate encryption (PE) scheme has the exact same interface as ABE, except that it requires some form of hiding of the attributes. In particular, in the presence of keys that do not decrypt, namely keys for functions f such that $f(x) = 0$, we require that the attributes x be completely hidden. However, if the adversary obtains the key for a function f such that $f(x) = 1$, we require no hiding. This is called *weak attribute hiding*. Thus, predicate encryption is just another way of saying “ABE with weak attribute hiding”.
- **Functional Encryption:** In a functional encryption scheme, one encrypts an input x and given a functional key sk_f , one can compute $f(x)$ and only $f(x)$. By a simple transformation, this turns out to be equivalent to an ABE with *strong attribute hiding*. Namely, where the encryptions of (x_0, M_0) and (x_1, M_1) are computationally indistinguishable given many functional keys sk_f where either: (a) $f(x_0) = f(x_1) = 0$; or (b) $f(x_0) = f(x_1) = 1$ and $M_0 = M_1$. Thus, functional encryption is just another way of saying “ABE with strong attribute hiding”.

A major contribution of our project was the systematic study of various types of functional encryption schemes. Our results made dramatic advances in the state of the art for functional encryption, constructing the first ABE schemes for general circuits [GVW13] and the first single-key succinct functional encryption for general circuits [GKP⁺13]. We describe our results in detail below. Figure 4.2 contains a summary of the most significant results to date in the field of functional encryption.

4.2.1 Attribute-based Encryption

In an attribute-based encryption (ABE) scheme, a ciphertext is associated with an L-bit public index IND and a message m , and a secret key is associated with a Boolean predicate P . The secret key allows to decrypt the ciphertext and learn m iff $P(IND)=1$. Moreover, the scheme should be secure against collusions of users, namely, given secret keys for polynomially many predicates, an adversary learns nothing about the message if none of the secret keys can individually decrypt the ciphertext.

Scheme	Class of Functions Supported	Assumption
Identity-based Encryption [BF01]	Point Functions	Bilinear DDH
Identity-based Encryption [Coc01]	Point Functions	Quadratic Residuosity
Fuzzy IBE [SW05]	Hamming distance	Bilinear DDH
Formula ABE [GPSW06]	Boolean formulas	Bilinear DDH
Circuit ABE [GVW13]	Circuits*	(Sub-exponential) LWE
(Compact) Circuit ABE [BGG ⁺ 14]	Circuits*	(Sub-exponential) LWE
Inner product PE [AFV11]	Inner product zero-testing	LWE
Circuit PE [GVW15]	Circuits*	(Sub-exponential) LWE
Inner product FE [KSW13, LOS ⁺ 10]	Inner product zero-testing	Bilinear [†]
Single-key FE [SS10]	Circuits	Any public-key encryption
Bounded-key FE [GVW12]	Circuits	Any public-key encryption
Bounded-key Succinct FE [GKP ⁺ 13]	Circuits*	(Sub-exponential) LWE
Many-key FE [GGH ⁺ 13]	Circuits	Existence of IO Obfuscation
Multi-input FE [GGG ⁺ 14]	Circuits	Existence of IO Obfuscation
Many-key FE [GGHZ14]	Circuits	Multi-linear Elimination

Figure 2: Attribute-based, Predicate and Functional Encryption Schemes. The first six rows are ABE, the next two PE and the rest are FE schemes. The rows in gray represent contributions from our team.

We present attribute-based encryption schemes for circuits of any arbitrary polynomial size, where the public parameters and the ciphertext grow linearly with the depth of the circuit. Our construction is secure under the standard learning with errors (LWE) assumption. Previous constructions of attribute-based encryption were for Boolean formulas, captured by the complexity class NC1.

In the course of our construction, we present a new framework for constructing ABE schemes. As a by-product of our framework, we obtain ABE schemes for polynomial-size branching programs, corresponding to the complexity class LOGSPACE, under quantitatively better assumptions.

4.2.2 Bounded-Key Functional Encryption

We construct a functional encryption scheme secure against an a priori bounded polynomial number of collusions for the class of all polynomial-size circuits. Our constructions require only semantically secure public-key encryption schemes and pseudo-random generators computable by small-depth circuits (known to be implied by most concrete intractability assumptions). For certain special cases such as predicate encryption schemes with public index, the construction requires only semantically secure encryption schemes, which is clearly the minimal necessary assumption.

Our constructions rely heavily on techniques from secure multiparty computation and randomized encodings. All our constructions are secure under a strong, adaptive simulation-based definition of functional encryption.

4.2.3 Succinct Functional Encryption and Reusable Garbled Circuits

Functional Encryption is a new paradigm for public-key encryption that enables fine-grained control of access to encrypted data. It provides, for instance, the ability to release secret keys associated with a keyword that can decrypt only those documents that contain the keyword. More generally, functional encryption

allows the owner of a “master” secret key to release restricted secret keys that reveal a specific function of encrypted data. This stands in stark contrast to traditional encryption, where access to the encrypted data is all or nothing.

Goldwasser, Vaikuntanathan and their co-authors in [GKP⁺13] proposed new functional encryption schemes which can evaluate functions and more generally run algorithms, over encrypted data in time which grows proportionally with the time it takes to evaluate the algorithms over the unencrypted data. This is in contrast with previous functional encryption schemes where the cost of running algorithms over encrypted data was proportional to the worst-case running time (over all possible data).

A circuit garbling scheme, which has been one of the most useful primitives in modern cryptography, is a construction originally suggested by Yao in the 80s in the context of secure two-party computation [Yao86]. This construction relies on the existence of a one-way function to encode an arbitrary circuit C (“garbling” the circuit) and then encode any input x to the circuit (where the size of the encoding is short, namely, it does not grow with the size of the circuit C); a party given the garbling of C and the encoding of x can run the garbled circuit on the encoded x and obtain $C(x)$. The most basic properties of garbled circuits are circuit and input privacy: an adversary learns nothing about the circuit C or the input x other than the result $C(x)$. Over the years, garbled circuits and variants thereof have found many applications. However, a basic limitation of the original construction remains: it offers only one-time usage. Specifically, providing an encoding of more than one input compromises the secrecy of the circuit. Thus, evaluating the circuit C on any new input requires an entirely new garbling of the circuit.

The problem of reusing garbled circuits has been open for 30 years. Using our newly constructed succinct functional encryption scheme we are now able to build reusable garbled circuits that achieve circuit and input privacy: a garbled circuit for any computation of depth d (where the parameters of the scheme depend on d), which can be run on any polynomial number of inputs without compromising the privacy of the circuit or the input.

4.2.4 Multi-Input Functional Encryption

We introduce the problem of Multi-Input Functional Encryption, where a secret key SK_f can correspond to an n -ary function f that takes multiple ciphertexts as input. Multi-input functional encryption is a general tool for computing on encrypting data which allows for mining aggregate information from several different data sources (rather than just a single source as in single input functional encryption). We show wide applications of this primitive to running SQL queries over encrypted database, non-interactive differentially private data release, delegation of computation, etc.

We formulate both indistinguishability-based and simulation-based definitions of security for this notion, and show close connections with indistinguishability and virtual black-box definitions of obfuscation. Assuming indistinguishability obfuscation for circuits, we present constructions achieving indistinguishability security for a large class of settings. We show how to modify this construction to achieve simulation-based security as well, in those settings where simulation security is possible. Assuming differing-inputs obfuscation [Barak et al., FOCS’01], we also provide a construction with similar security guarantees as above, but where the keys and ciphertexts are compact.

4.3 Results: Large-Scale Multiparty Computation

It turns out that methods for computing on “encrypted” data, broadly defined, were known even before the current buzz on fully homomorphic encryption. In particular, the notion of secure multi-party computation

(MPC) has been thoroughly studied over the past decades. Secure multiparty computation protocols allow us $n \geq 2$ parties, each holding its own input, to collaborate and compute a joint function of their inputs, while ensuring that each party reveals nothing else to the others.

The vast majority of works assume a full communication pattern: every party exchanges messages with all the network participants over a complete network of point-to-point channels. This can be problematic in modern large scale networks, where the number of parties can be of the order of millions, as for example when computing on large distributed data.

Motivated by the above observation, PI Goldwasser, together with graduate student Elette Boyle and postdoc Stefano Tessaro [BGT13] put forward the notion of communication locality, namely, the total number of point-to-point channels that each party uses in the protocol, as a quality metric of MPC protocols. They proved that assuming a public-key infrastructure (PKI) and a common reference string (CRS), an MPC protocol can be constructed for computing any n -party function, with the following properties:

- The communication locality of the protocol is $O(\log^c n)$, for an appropriate constant c . that is each party need only talk to $O(\log^c n)$ other parties;
- The round complexity $O(\log^{c'} n)$, for an appropriate constant c' ;
- The protocol tolerates a static (i.e., non-adaptive) adversary corrupting up to $t < (1/3 - \epsilon)n$ parties for any given constant $0 < \epsilon < 1/3$.

Continuing along this line of thought, further work of PI Goldwasser together with collaborators addressed two questions left open by this result: (a) Can we achieve low communication locality and round complexity while tolerating adaptive adversaries? and (b) Can we achieve low communication locality with optimal resiliency $t < n/2$?

In work with collaborators [CCG⁺15], PI Goldwasser answered both questions affirmatively. First, we consider the model from [BGT13], where we replace the CRS with a symmetric-key infrastructure (SKI). In this model we give a protocol with communication locality and round complexity $\text{polylog}(n)$ (as in the work of [BGT13]) which tolerates up to $t < n/2$ adaptive corruptions, under a standard intractability assumption for adaptively secure protocols, namely, the existence of trapdoor permutations whose domain has invertible sampling. This is done by using the SKI to derive a sequence of random hidden communication graphs among players. A central new technique then shows how to use these graphs to emulate a complete network in $\text{polylog}(n)$ rounds while preserving the $\text{polylog}(n)$ locality. Second, we show how we can even remove the SKI setup assumption at the cost, however, of increasing the communication locality (but not the round complexity) by a factor of \sqrt{n} .

4.4 Results: Leakage-Resilient Computation

The absolute privacy of the secret keys associated with cryptographic algorithms has been the corner-stone of modern cryptography. Modern cryptographic algorithms are designed under the assumption that keys are perfectly secret, and computations done within your personal computer seem like a black-box to the outside. Still, in practice, keys do get compromised at times and computations are not opaque for a variety of reasons. A particularly disturbing loss of secrecy is as a result of *side channel attacks* (see [Koc96, KJJ99, QS01, AARR02, QK02, BE03, Rel, ECR] for many examples). These attacks exploit the fact that every cryptographic algorithm is ultimately implemented on a physical device and such implementations enable “observations” which can be made and measured on secret data and secret keys. Indeed, side channel

observations can lead to information leakage about secret keys, which in turn can and have lead to complete breaks of systems which have been proved mathematically secure, without violating any of the underlying mathematical principles or assumptions. Traditionally, such attacks have been followed by ad-hoc “fixes” which make specific implementation invulnerable to particular attacks, only to potentially be broken again by new examples of side-channel attacks.

In recent years, starting with the works of Canetti, Dodis, Halevi, Kushilevitz and Sahai [CDH⁺00], Ishai, Sahai and Wagner [ISW03] and Micali and Reyzin [MR04], a new goal has been set within the cryptography community: *to build general theories of physical security against large classes of families of side channel attacks*. A large body of work has accumulated by now [CDH⁺00, DSS01, ISW03, MR04, DP08, AGV09, ADW09, NS09, DKL09, Pie09, PSP⁺08, GKR08, DP08] in which different classes of side channel attacks have been defined and different cryptographic primitives have been designed to provably withstand these attacks.

Any cryptographic protocol, including decryption and signature algorithms, multi-party computation and zero-knowledge, is potentially vulnerable to leakage, and measures must be taken to protect them. Our goal is to design general purpose tools that accomplish this objective.

4.4.1 Goldwasser-Rothblum Leakage-Resilience Compiler

Goldwasser and Rothblum [GR07] address the following problem: how to execute any algorithm P , for an unbounded number of executions, in the presence of an adversary who observes partial information on the internal state of the computation during executions. The security guarantee is that the adversary learns nothing, beyond P 's input/output behavior. This general problem is important for running cryptographic algorithms in the presence of side-channel attacks, as well as for running non-cryptographic algorithms, such as a proprietary search algorithm or a game, on a cloud server where parts of the execution's internals might be observed.

Their main result is a compiler, which takes as input an algorithm P and a security parameter k , and produces a functionally equivalent algorithm P' such that the running time of P' is a factor of $\text{poly}(n)$ slower than P and is composed of a series of calls to $\text{poly}(n)$ time computable sub-algorithms. During the executions of P' , an adversary algorithm A which can choose the inputs of P' and can learn the results of adaptively chosen leakage functions, each of bounded output size $\Omega(n)$, on the sub-algorithms of P' and the randomness they use.

They show that for any computationally unbounded A observing the results of computationally unbounded leakage functions, will learn no more from its observations than it could given black-box access only to the input-output behavior of P . This result is unconditional and does not rely on any secure hardware components.

4.4.2 Leakage-Resilient Multiparty Computation

The problem of leakage on secret inputs is applicable to multi-party secure function evaluation as well as the basic cryptographic primitives. For some multi-party functions, such as voting, such leakage can be detrimental.

In work of Goldwasser et. al. [BGJK12], multiparty computation (MPC) protocols are constructed that are secure even if a malicious adversary which, in addition to corrupting constant $1 > c > 0$ fraction of parties, can leak information about the secret state of each *honest, non-corrupt* party. This leakage can be continuous for an unbounded number of executions of the MPC protocol, computing different functions on the same or different set of inputs. We assumed a (necessary) “leak-free” preprocessing stage.

We emphasize that leakage resilience is achieved without weakening the security guarantee of classical MPC. Namely, an adversary who is given leakage on honest parties' states, is guaranteed to learn nothing beyond the input and output values of corrupted parties. Our result relies on standard cryptographic assumptions, and our security parameter is polynomially related to the number of parties.

4.5 Results: Functional Signatures and Pseudorandom Functions

In the spirit of our work on functional encryption, PI Goldwasser together with graduate students Boyle and Ivan [BGI14a] introduced two new cryptographic primitives named functional signature and functional pseudorandom functions (PRFs) and showed how to construct them under some cryptographic hardness assumptions. In a functional signature scheme, in addition to a master signing key, there are also auxiliary signing keys each defined per different function f , which allow one to sign message M if and only if $f(m) = 1$. In a functional PRF, there are auxiliary secret keys each for a function f , which allow one to evaluate PRF on any y if $f(y) = 1$.

A natural application of functional signature schemes is the delegation of the signing process: For a policy P , consider a function f such that $f(m) = 1$ if $P(m) = 1$ and $f(m) = \text{NIL}$ otherwise; then a signing key for f allows one to sign message m if and only if $P(m) = 1$. Similarly, with functional PRFs, one can construct PRFs with selective access, in which there are keys for a policy P that allow one to evaluate PRF on any x if $P(x) = 1$.

Another application of functional signatures is to certify that only allowable computations were performed on data. For example, imagine the setting of a digital camera that produces signed photos (i.e. the original photos produced by the camera can be certified). In this case, one may want to allow photo-processing software to perform minor touch-ups of the photos, such as changing the color scale or removing red-eyes, but not allow more significant changes such as merging two photos or cropping a picture. But, how can an original photo which is slightly touched-up be distinguished from one which is the result of a major change? Functional signatures can naturally address this problem by providing the photo processing software with keys which enable it to sign only the allowable modifications of an original photograph. Generalizing, we think of a client and a server (e.g. photo-processing software), where the client provides the server with data (e.g. signed original photos, text documents, medical data) which he wants to be processed in a restricted fashion. A functional signature of the processed data provides proof of allowable processing.

4.5.1 Functional Signatures and Pseudorandom Functions

Pseudorandom functions, introduced by Goldreich, Goldwasser, and Micali in 1986 [GGM86], are a family of indexed functions $F = \{F_s\}$ such that: (1) given the index s , F_s can be efficiently evaluated on all inputs (2) no probabilistic polynomial-time algorithm without s can distinguish evaluations $F_s(x_i)$ for inputs x_i chosen adversarially from random values. Pseudorandom functions are useful for numerous symmetric-key cryptographic applications, including generating passwords, identify-friend-or-foe systems, and symmetric-key encryption secure against chosen ciphertext attacks. In the aforementioned work, Goldwasser et al [BGI14b] extend pseudorandom functions to a primitive which they call functional pseudorandom functions (F-PRF). The idea is that in addition to a master secret key (that can be used to evaluate the pseudorandom function F_s on any point in the domain), there are additional secret keys skf per function f , which allow one to evaluate F_s on any y for which there exists x such that $f(x) = y$ (i.e. $y \in \text{Range}(f)$). An immediate application of such a construct is to specify succinctly the randomness to be used by parties in a randomized distributed protocol with potentially faulty players, so as to force honest behavior.

The notion of functional pseudorandom functions has many variations. One natural variant that immediately follows is pseudorandom functions with selective access: start with a pseudorandom function as defined in [GGM86], and add the ability to generate secondary keys sk_{Pi} (per predicate P_i) which enable computing $F_s(x)$ whenever $P_i(x) = 1$. This is a special case of F-PRF, as we can take the secret key for predicate P_i to be sk_{fi} where $f_i(x) = x$ if $P_i(x) = 1$ and NIL otherwise. The special case of punctured PRFs, in which secondary keys allow computing $F_s(x)$ on all inputs except one, is similarly implied and has recently been shown to have important applications (e.g., [SW14]). (We note that independently of our work, a similar primitive was introduced by Bohen et al and named constrained PRF)

4.5.2 Constrained PRFs for Arbitrary Circuits from LWE

Boneh et al. (Crypto 13) and Banerjee and Peikert (Crypto 14) constructed pseudorandom functions (PRFs) from the Learning with Errors (LWE) assumption by embedding combinatorial objects, a path and a tree respectively, in instances of the LWE problem. In this work, we show how to generalize this approach to embed *circuits*, inspired by recent progress in the study of Attribute Based Encryption.

Embedding a universal circuit for some class of functions allows us to produce *constrained keys* for functions in this class, which gives us the first standard-lattice-assumption-based constrained PRF (CPRF) for general bounded-description bounded-depth functions, for arbitrary polynomial bounds on the description size and the depth. (A constrained key w.r.t a circuit C enables one to evaluate the PRF on all x for which $C(x) = 1$, but reveals nothing on the PRF values at other points.) We rely on the LWE assumption and on the one-dimensional SIS (Short Integer Solution) assumption, which are both related to the worst case hardness of general lattice problems. Previous constructions for similar function classes relied on such exotic assumptions as the existence of multilinear maps or secure program obfuscation. The main drawback of our construction is that it does not allow collusion (i.e. to provide more than a single constrained key to an adversary). Similarly to the aforementioned previous works, our PRF family is also *key homomorphic*.

Interestingly, our constrained keys are very short. Their length does not depend directly either on the size of the constraint circuit or on the input length. We are not aware of any prior construction achieving this property, even relying on strong assumptions such as indistinguishability obfuscation.

4.5.3 Aggregate Pseudo-random Functions and Connections to Learning Theory

In the first part of this work, we introduce a new type of pseudo-random function for which “aggregate queries” over exponential-sized sets can be efficiently answered. An example of an aggregate query may be the product of all function values belonging to an exponential-sized interval, or the sum of all function values on points for which a polynomial time predicate holds. We show how to use algebraic properties of underlying classical pseudo random functions, to construct aggregatable pseudo random functions for a number of classes of aggregation queries under cryptographic hardness assumptions. On the flip side, we show that certain aggregate queries are impossible to support.

In the second part of this work, we show how various extensions of pseudo-random functions considered recently in the cryptographic literature, yield impossibility results for various extensions of machine learning models, continuing a line of investigation originated by Valiant and Kearns in the 1980s and 1990s. The extended pseudo-random functions we address include constrained pseudo random functions, aggregatable pseudo random functions, and pseudo random functions secure under related-key attacks.

Conclusions and Recommendations

We conclude by addressing the question of which FHE scheme to use in practice.

It appears that the BGV scheme [BGV12] or one of its adaptations (following Brakerski's scale-invariant technique [Bra12], SIMD techniques [GHS12a], using an NTRU variant [LTV12, BLLN13]) is the method of choice to get the maximal efficiency in practice. Indeed, the homomorphic encryption library HE-Lib [HS15] implements the BGV encryption scheme + the GHS SIMD techniques. Although the third generation of FHE schemes appears to be more efficient, they lack mechanisms for SIMD evaluation at the time of writing. This remains a major open problem in the field, with potentially dramatic implications for efficiency.

Regarding the ABE, functional encryption and other advanced schemes, the efficiency of these schemes has been steadily improving over the years, and we believe the current schemes specialized to particular functions should be efficient enough in practice. Regardless, we leave implementations of these schemes as future work.

Bibliography

- [AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, August 13-15 2002.
- [ACM88] *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, Chicago, Illinois, 2–4 May 1988.
- [ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Halevi [Hal09], pages 36–54.
- [AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 21–40. Springer, 2011.
- [AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *Sixth Theory of Cryptography Conference — TCC 2007*, volume 5444 of *Lecture Notes in Computer Science*. Springer-Verlag, 2009.
- [AP14] Jacob Alperin-Sheriff and Chris Peikert. Faster bootstrapping with polynomial error. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 297–314. Springer, 2014.
- [BE03] Hagai Bar-El. Known attacks against smartcards, 2003. http://www.hbareil.com/publications/Known_Attacks_Against_Smartcards.pdf, last accessed: August 26, 2009.
- [BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology—CRYPTO 2001*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.
- [BGG⁺14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications*

of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. *Proceedings*, pages 533–556, 2014.

- [BGI14a] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 501–519, 2014.
- [BGI14b] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519. Springer, 2014.
- [BGJK12] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual memory leakage. In Karloff and Pitassi [KP12], pages 1235–1254.
- [BGT13] Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multiparty computation - how to run sublinear algorithms in a distributed setting. In *TCC*, pages 356–376, 2013.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 309–325. ACM, 2012.
- [BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In ACM [ACM88], pages 1–10.
- [BLLN13] Joppe W. Bos, Kristin E. Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *Cryptography and Coding - 14th IMA International Conference, IMACC 2013, Oxford, UK, December 17-19, 2013. Proceedings*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2013.
- [BPTG15] Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014*. The Internet Society, 2015.
- [Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Safavi-Naini and Canetti [SC12], pages 868–886.
- [BRF13] Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors. *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*. ACM, 2013.
- [BSW12] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Commun. ACM*, 55(11):56–64, 2012.

- [BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Ostrovsky [Ost11], pages 97–106.
- [BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 505–524. Springer, 2011.
- [BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based FHE as secure as PKE. In Moni Naor, editor, *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, pages 1–12. ACM, 2014.
- [CCG⁺15] Nishanth Chandran, Wutichai Chongchitmate, Juan A. Garay, Shafi Goldwasser, Rafail Ostrovsky, and Vassilis Zikas. The hidden graph model: Communication locality and optimal resiliency with adaptive faults. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 153–162. ACM, 2015.
- [CDH⁺00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 453–469. Springer-Verlag, 2000.
- [CM15] Michael Clear and Ciaran McGoldrick. Multi-identity and multi-key leveled FHE from learning with errors. In Gennaro and Robshaw [GR15], pages 630–656.
- [Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363. Springer, 2001.
- [DDS14] Wei Dai, Yarkin Doröz, and Berk Sunar. Accelerating NTRU based homomorphic encryption using gpus. In *IEEE High Performance Extreme Computing Conference, HPEC 2014, Waltham, MA, USA, September 9-11, 2014*, pages 1–6. IEEE, 2014.
- [DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *STOC*, pages 621–630. ACM, 2009.
- [DM15] Léo Ducas and Daniele Micciancio. FHEW: bootstrapping homomorphic encryption in less than a second. In Oswald and Fischlin [OF15], pages 617–640.
- [DÖSS15] Yarkin Doröz, Erdiñç Öztürk, Erkey Savas, and Berk Sunar. Accelerating LTV based homomorphic encryption in reconfigurable hardware. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 185–204. Springer, 2015.

- [DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Symposium on Foundations of Computer Science*, pages 293–302, Philadelphia, PA, USA, October 25–28 2008. IEEE Computer Society.
- [DSS01] Yevgeniy Dodis, Amit Sahai, and Adam Smith. On perfect and adaptive security in exposure-resilient cryptography. In Birgit Pfitzmann, editor, *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 301–324. Springer-Verlag, 2001.
- [ECR] ECRYPT. Side channel cryptanalysis lounge. http://www.crypto.ruhr-uni-bochum.de/en_sclounge.html, last accessed: August 26, 2009.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 169–178. ACM, 2009.
- [GGG⁺14] Shafi Goldwasser, S. Dov Gordon, Vipul Goyal, Abhishek Jain, Jonathan Katz, Feng-Hao Liu, Amit Sahai, Elaine Shi, and Hong-Sheng Zhou. Multi-input functional encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 578–602. Springer, 2014.
- [GGH⁺13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 40–49. IEEE Computer Society, 2013.
- [GGHZ14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mark Zhandry. Fully secure functional encryption without obfuscation. *IACR Cryptology ePrint Archive*, 2014:666, 2014.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. Extended abstract in FOCS 84.
- [GH11] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Ostrovsky [Ost11], pages 107–109.
- [GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.
- [GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. In Safavi-Naini and Canetti [SC12], pages 850–867.
- [Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

- [GKP⁺13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Boneh et al. [BRF13], pages 555–564.
- [GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *STOC*, pages 113–122. ACM, 2008.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, 25–27 May 1987.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*, pages 89–98. ACM, 2006.
- [GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 194–213. Springer, 2007.
- [GR15] Rosario Gennaro and Matthew Robshaw, editors. *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, volume 9216 of *Lecture Notes in Computer Science*. Springer, 2015.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer, 2013.
- [GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In Safavi-Naini and Canetti [SC12], pages 162–179.
- [GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Boneh et al. [BRF13], pages 545–554.
- [GVW15] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Predicate encryption for circuits from LWE. In Gennaro and Robshaw [GR15], pages 503–523.
- [Hal09] Shai Halevi, editor. *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*. Springer-Verlag, 2009.
- [HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. NTRU: A ring-based public key cryptosystem. In Joe Buhler, editor, *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.

- [HS15] Shai Halevi and Victor Shoup. Bootstrapping for helib. In Oswald and Fischlin [OF15], pages 641–670.
- [ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology—CRYPTO 2003*, volume 2729 of *LNCS*. Springer-Verlag, 2003.
- [KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology—CRYPTO '99*, volume 1666 of *LNCS*, pages 388–397. Springer-Verlag, 15–19 August 1999.
- [Koc96] Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 18–22 August 1996.
- [KP12] Howard J. Karloff and Toniann Pitassi, editors. *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*. ACM, 2012.
- [KSW13] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *J. Cryptology*, 26(2):191–224, 2013.
- [LOS⁺10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Gilbert [Gil10], pages 62–91.
- [LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Karloff and Pitassi [KP12], pages 1219–1234.
- [MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *First Theory of Cryptography Conference — TCC 2004*, volume 2951 of *LNCS*, pages 278–296. Springer-Verlag, February 19–21 2004.
- [MW15] Pratyay Mukherjee and Daniel Wichs. Two round MPC from LWE via multi-key FHE. *IACR Cryptology ePrint Archive*, 2015:345, 2015.
- [NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Halevi [Hal09], pages 18–35.
- [OF15] Elisabeth Oswald and Marc Fischlin, editors. *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*. Springer, 2015.
- [Ost11] Rafail Ostrovsky, editor. *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. IEEE Computer Society, 2011.
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342, 2009.

- [Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*, pages 462–482. Springer-Verlag, 2009.
- [PSP⁺08] Christophe Petit, François-Xavier Standaert, Olivier Pereira, Tal Malkin, and Moti Yung. A block cipher based pseudo random number generator secure against side-channel key recovery. In Masayuki Abe and Virgil D. Gligor, editors, *ASIACCS*, pages 56–65. ACM, March 18-20 2008.
- [QK02] Jean-Jaques Quisquater and François Koene. Side channel attacks: State of the art, October 2002. http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf, last accessed: August 26, 2009.
- [QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *E-smart*, volume 2140 of *LNCS*, pages 200–210. Springer-Verlag, September 19-21 2001.
- [RC14] Kurt Rohloff and David Bruce Cousins. A scalable implementation of fully homomorphic encryption built on NTRU. In Rainer Böhme, Michael Brenner, Tyler Moore, and Matthew Smith, editors, *Financial Cryptography and Data Security - FC 2014 Workshops, BITCOIN and WAHC 2014, Christ Church, Barbados, March 7, 2014, Revised Selected Papers*, volume 8438 of *Lecture Notes in Computer Science*, pages 221–234. Springer, 2014.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.
- [Rel] Reliable Computing Laboratory, Boston University. Side channel attacks database. <http://www.sidechannelattacks.com>, last accessed: August 26, 2009.
- [SC12] Reihaneh Safavi-Naini and Ran Canetti, editors. *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012.
- [SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 463–472. ACM, 2010.
- [SV09] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. *IACR Cryptology ePrint Archive*, 2009:571, 2009.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *Advances in Cryptology—EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 457–473. Springer-Verlag, 2005.
- [SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

- [vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [Gil10], pages 24–43.
- [Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science, Toronto, Canada, 27-29 October 1986*, pages 162–167. IEEE Computer Society, 1986.

Appendix - Published Papers

Efficient Fully Homomorphic Encryption from (Standard) LWE

Zvika Brakerski*

Vinod Vaikuntanathan†

Abstract

We present a fully homomorphic encryption scheme that is based solely on the (standard) learning with errors (LWE) assumption. Applying known results on LWE, the security of our scheme is based on the worst-case hardness of “short vector problems” on arbitrary lattices.

Our construction improves on previous works in two aspects:

1. We show that “somewhat homomorphic” encryption can be based on LWE, using a new *re-linearization* technique. In contrast, all previous schemes relied on complexity assumptions related to ideals in various rings.
2. We deviate from the “squashing paradigm” used in all previous works. We introduce a new *dimension-modulus reduction* technique, which shortens the ciphertexts and reduces the decryption complexity of our scheme, *without introducing additional assumptions*.

Our scheme has very short ciphertexts and we therefore use it to construct an asymptotically efficient LWE-based single-server private information retrieval (PIR) protocol. The communication complexity of our protocol (in the public-key model) is $k \cdot \text{polylog}(k) + \log |\mathbf{DB}|$ bits per single-bit query (here, k is a security parameter).

*Weizmann Institute of Science. Email: zvika.brakerski@weizmann.ac.il. The author’s research was supported by ISF grant 710267, BSF grant 710613, and NSF contracts CCF-1018064 and CCF-0729011.

†University of Toronto. Email: vinodv@cs.toronto.edu.

1 Introduction

Fully-homomorphic encryption is one of the holy grails of modern cryptography. In a nutshell, a fully homomorphic encryption scheme is an encryption scheme that allows evaluation of arbitrarily complex programs on encrypted data. The problem was suggested by Rivest, Adleman and Dertouzos [RAD78] back in 1978, yet the first plausible candidate came thirty years later with Gentry’s breakthrough work in 2009 [Gen09b, Gen10] (although, there has been partial progress in the meanwhile [GM82, Pai99, BGN05, IP07]).

Gentry’s work showed for the first time that fully homomorphic encryption can be based on cryptographic assumptions. However, his solution involved new and relatively untested cryptographic assumptions. Our work aims to put fully homomorphic encryption on standard, well-studied cryptographic assumptions.

The main building block in Gentry’s construction (a so-called “somewhat” homomorphic encryption scheme) was based on the (worst-case, quantum) hardness of problems on *ideal lattices*.¹ Although lattices have become standard fare in cryptography and lattice problems have been relatively well-studied, ideal lattices are a special breed that we know relatively little about. Ideals are a natural mathematical object to use to build fully homomorphic encryption in that they natively support both addition and multiplication (whereas lattices are closed under addition only). Indeed, all subsequent constructions of fully homomorphic encryption [SV10, DGHV10, BV11] relied on ideals in various rings in an explicit way. Our first contribution is the construction of a “somewhat” homomorphic encryption scheme whose security relies solely on the (worst-case, classical) hardness of standard problems on *arbitrary* (not necessarily ideal) *lattices*.

Secondly, in order to achieve *full* homomorphism, Gentry had to go through a so-called “squashing step” which forced him to make an additional very strong hardness assumption – namely, the hardness of the (average-case) sparse subset-sum problem. As if by a strange law of nature, all the subsequent solutions encountered the same difficulty as Gentry did in going from a “somewhat” to a fully homomorphic encryption, and they all countered this difficulty by relying on the same sparse subset-sum assumption. This additional assumption was considered to be the main caveat of Gentry’s solution and removing it has been, perhaps, the main open problem in the design of fully homomorphic encryption schemes. Our second contribution is to remove the necessity of this additional assumption.

Thus, in a nutshell, we construct a fully homomorphic encryption scheme whose security is based solely on the classical hardness of solving standard lattice problems in the worst-case.² Specifically, our scheme is based on the learning with errors (LWE) assumption that is known to be at least as hard as solving hard problems in general lattices. Thus our solution does not rely on lattices directly and is fairly natural to understand and implement.

To achieve our goals, we deviate from two paradigms that ruled the design of (a handful of) candidate fully homomorphic encryption schemes [Gen09b, SV10, DGHV10, BV11]:

1. We introduce the *re-linearization* technique, and show how to use it to obtain a *somewhat* homomorphic encryption that does not require hardness assumptions on *ideals*.

¹Roughly speaking, ideal lattices correspond to a geometric embedding of an ideal in a number field. See [LPR10] for a precise definition.

²Strictly speaking, under this assumption, our scheme can evaluate polynomial-size circuits with a-priori bounded (but arbitrary) depth. A fully homomorphic encryption scheme independent of the circuit depth can be obtained by making an additional “circular security” assumption. See Section 3.

2. We present a *dimension-modulus reduction* technique, that turns our somewhat homomorphic scheme into a fully homomorphic one, without the need for the artificial *squashing* step and the sparse subset-sum assumption.

We provide a detailed overview of these new techniques in Sections 1.1, 1.2 below.

Interestingly, the ciphertexts of the resulting fully homomorphic scheme are very short. This is a desirable property which we use, in conjunction with other techniques, to achieve very efficient private information retrieval protocols. See also Section 1.3 below.

1.1 Re-Linearization: Somewhat Homomorphic Encryption without Ideals

The starting point of Gentry’s construction is a “somewhat” homomorphic encryption scheme. For a class of circuits \mathcal{C} , a \mathcal{C} -homomorphic scheme is one that allows evaluation of any circuit in the class \mathcal{C} . The simple, yet striking, observation in Gentry’s work is that if a (slightly augmented) decryption circuit for a \mathcal{C} -homomorphic scheme resides in \mathcal{C} , then the scheme can be converted (or “bootstrapped”) into a fully homomorphic encryption scheme.

It turns out that encryption schemes that can evaluate a non-trivial number of addition and multiplication operations³ are already quite hard to come by (even without requiring that they are bootstrappable).⁴ Gentry’s solution to this was based on the algebraic notion of *ideals* in rings. In a very high level, the message is considered to be a ring element, and the ciphertext is the message masked with some “noise”. The novelty of this idea is that the noise itself belonged to an ideal I . Thus, the ciphertext is of the form $m + xI$ (for some x in the ring). Observe right off the bat that the scheme is born additively homomorphic; in fact, that will be the case with all the schemes we consider in this paper. The ideal I has two main properties: first, a random element in the ideal is assumed to “mask” the message; and second, it is possible to generate a secret trapdoor that “annihilates” the ideal, i.e., implementing the transformation $m + xI \rightarrow m$. The first property guarantees security, while the second enables multiplying ciphertexts. Letting c_1 and c_2 be encryptions of m_1 and m_2 respectively,

$$c_1 c_2 = (m_1 + xI)(m_2 + yI) = m_1 m_2 + (m_1 y + m_2 x + xyI)I = m_1 m_2 + zI$$

When decrypting, the ideal is annihilated and the product $m_1 m_2$ survives. Thus, $c_1 c_2$ is indeed an encryption of $m_1 m_2$, as required. This nifty solution required, as per the first property, a hardness assumption on ideals in certain rings. Gentry’s original work relied on hardness assumptions on *ideal lattices*, while van Dijk, Gentry, Halevi and Vaikuntanathan [DGHV10] presented a different instantiation that considered ideals over the integers.

Our somewhat homomorphic scheme is based on the hardness of the “learning with errors” (LWE) problem, first presented by Regev [Reg05]. The LWE assumption states that if $\mathbf{s} \in \mathbb{Z}_q^n$ is an n dimensional “secret” vector, any polynomial number of “noisy” random linear combinations of the coefficients of \mathbf{s} are computationally indistinguishable from uniformly random elements in \mathbb{Z}_q . Mathematically,

$$\{\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i\}_{i=1}^{\text{poly}(n)} \stackrel{c}{\approx} \{\mathbf{a}_i, u_i\}_{i=1}^{\text{poly}(n)},$$

³All known scheme, including ours, treat evaluated functions as arithmetic circuits. Hence we use the terminology of “addition and multiplication” gates. The conversion to the boolean model (AND, OR, NOT gates) is immediate.

⁴We must mention here that we are interested only in *compact* fully homomorphic encryption schemes, namely ones where the ciphertexts do not grow in size with each homomorphic operation. If we do allow such growth in size, a number of solutions are possible. See, e.g., [SY99, GHV10a, MGH10].

where $\mathbf{a}_i \in \mathbb{Z}_q^n$ and $u_i \in \mathbb{Z}_q$ are uniformly random, and the “noise” e_i is sampled from a noise distribution that outputs numbers much smaller than q (an example is a discrete Gaussian distribution over \mathbb{Z}_q with small standard deviation).

The LWE assumption does not refer to ideals, and indeed, the LWE problem is at least as hard as finding short vectors in *any lattice*, as follows from the worst-case to average-case reductions of Regev [Reg05] and Peikert [Pei09]. As mentioned earlier, we have a much better understanding of the complexity of lattice problems (thanks to [LLL82, Ajt98, Mic00] and many others), compared to the corresponding problems on ideal lattices. In particular, despite considerable effort, the best known algorithms to solve the LWE problem run in time nearly exponential in the dimension n .⁵ The LWE assumption also turns out to be particularly amenable to the construction of simple, efficient and highly expressive cryptographic schemes (e.g., [Reg05, GPV08, AGV09, ACPS09, CHKP10, ABB10] and many others). Our construction of a fully homomorphic encryption scheme from LWE is perhaps a very strong testament to its power and elegance.

Constructing a (secret-key) encryption scheme whose security is based on the LWE assumption is rather straightforward. To encrypt a bit $m \in \{0, 1\}$ using secret key $\mathbf{s} \in \mathbb{Z}_q^n$, we choose a random vector $\mathbf{a} \in \mathbb{Z}_q^n$ and a “noise” e and output the ciphertext

$$c = (\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$$

The key observation in decryption is that the two “masks” – namely, the secret mask $\langle \mathbf{a}, \mathbf{s} \rangle$ and the “even mask” $2e$ – do not interfere with each other.⁶ That is, one can decrypt this ciphertext by annihilating the two masks, one after the other: The decryption algorithm first re-computes the mask $\langle \mathbf{a}, \mathbf{s} \rangle$ and subtracts it from b , resulting in $2e + m \pmod{q}$. Since $e \ll q$, then $2e + m \pmod{q} = 2e + m$. Removing the even mask is now easy – simply compute $2e + m$ modulo 2.⁷

As we will see below, the scheme is naturally additive homomorphic, yet multiplication presents a thorny problem. In fact, a recent work of Gentry, Halevi and Vaikuntanathan [GHV10b] showed that (a slight variant of) this scheme supports *just a single* homomorphic multiplication, but at the expense of a huge blowup to the ciphertext which made further advance impossible.

To better understand the homomorphic properties of this scheme, let us shift our focus away from the encryption algorithm, on to the decryption algorithm. Given a ciphertext (\mathbf{a}, b) , consider the symbolic linear function $f_{\mathbf{a}, b} : \mathbb{Z}_q^n \rightarrow \mathbb{Z}_q$ defined as:

$$f_{\mathbf{a}, b}(\mathbf{x}) = b - \langle \mathbf{a}, \mathbf{x} \rangle \pmod{q} = b - \sum_{i=1}^n \mathbf{a}[i] \cdot \mathbf{x}[i] \in \mathbb{Z}_q$$

where $\mathbf{x} = (\mathbf{x}[1], \dots, \mathbf{x}[n])$ denotes the variables, and (\mathbf{a}, b) forms the public coefficients of the linear equation. Clearly, decryption of the ciphertext (\mathbf{a}, b) is nothing but evaluating this function on the secret key \mathbf{s} (and then taking the result modulo 2).⁸

⁵The nearly exponential time is for a large enough error (i.e., one that is a $1/\text{poly}(n)$ fraction of the modulus q). For smaller errors, as we will encounter in our scheme, there are better – but not significantly better – algorithms. In particular, if the error is a $1/2^{n^\epsilon}$ fraction of the modulus q , the best known algorithm runs in time approx. $2^{n^{1-\epsilon}}$.

⁶We remark that using $2e$ instead of e as in the original formulation of LWE does not adversely impact security, so long as q is odd (since in that case 2 is a unit in \mathbb{Z}_q).

⁷Although the simplified presentation of Gentry’s scheme above seems to deal with just one mask (the “secret mask”), in reality, the additional “even mask” existed in the schemes of [Gen09b, DGHV10] as well. Roughly speaking, they needed this to ensure semantic security, as we do.

⁸The observation that an LWE-based ciphertext can be interpreted as a linear equation of the secret was also used in [BV11].

Homomorphic addition and multiplication can now be described in terms of this function f . Adding two ciphertexts corresponds to the addition of two linear functions, which is again another linear function. In particular, $f_{(\mathbf{a}+\mathbf{a}',b+b')}(\mathbf{x}) = f_{\mathbf{a},b}(\mathbf{x}) + f_{\mathbf{a}',b'}(\mathbf{x})$ is the linear function corresponding to the “homomorphically added” ciphertext $(\mathbf{a} + \mathbf{a}', b + b')$. Similarly, multiplying two such ciphertexts corresponds to a symbolic multiplication of these linear equations

$$\begin{aligned} f_{(\mathbf{a},b)}(\mathbf{x}) \cdot f_{(\mathbf{a}',b')}(\mathbf{x}) &= (b - \sum \mathbf{a}[i]\mathbf{x}[i]) \cdot (b' - \sum \mathbf{a}'[i]\mathbf{x}[i]) \\ &= h_0 + \sum h_i \cdot \mathbf{x}[i] + \sum h_{i,j} \cdot \mathbf{x}[i]\mathbf{x}[j] , \end{aligned}$$

which results in a degree-2 polynomial in the variables $\mathbf{x} = (\mathbf{x}[1], \dots, \mathbf{x}[n])$, with coefficients $h_{i,j}$ that can be computed from (\mathbf{a}, b) and (\mathbf{a}', b') by opening parenthesis of the expression above. Decryption, as before, involves evaluating this quadratic expression on the secret key \mathbf{s} (and then reducing modulo 2). We now run into a serious problem – the decryption algorithm has to know all the coefficients of this quadratic polynomial, which means that the size of the ciphertext just went up from $n + 1$ elements to (roughly) $n^2/2$.

This is where our re-linearization technique comes into play. Re-linearization is a way to reduce the size of the ciphertext back down to $n + 1$. The main idea is the following: imagine that we publish “encryptions” of all the linear and quadratic terms in the secret key \mathbf{s} , namely all the numbers $\mathbf{s}[i]$ as well as $\mathbf{s}[i]\mathbf{s}[j]$, under a new secret key \mathbf{t} . Thus, these ciphertexts (for the quadratic terms) look like $(\mathbf{a}_{i,j}, b_{i,j})$ where

$$b_{i,j} = \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle + 2e_{i,j} + \mathbf{s}[i] \cdot \mathbf{s}[j] \approx \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle + \mathbf{s}[i] \cdot \mathbf{s}[j] .^9$$

Now, the sum $h_0 + \sum h_i \cdot \mathbf{s}[i] + \sum h_{i,j} \cdot \mathbf{s}[i]\mathbf{s}[j]$ can be written (approximately) as

$$h_0 + \sum h_i(b_i - \langle \mathbf{a}_i, \mathbf{t} \rangle) + \sum_{i,j} h_{i,j} \cdot (b_{i,j} - \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle) ,$$

which, lo and behold, is a linear function in \mathbf{t} ! The bottom-line is that multiplying the two linear functions $f_{(\mathbf{a},b)}$ and $f_{(\mathbf{a}',b')}$ and then re-linearizing the resulting expression results in a linear function (with $n + 1$ coefficients), whose evaluation on the new secret key \mathbf{t} results in the product of the two original messages (upon reducing modulo 2). The resulting ciphertext is simply the coefficients of this linear function, of which there are at most $n + 1$. This ciphertext will decrypt to $m \cdot m'$ using the secret key \mathbf{t} .

In this semi-formal description, we ignored an important detail which has to do with the fact that the coefficients $h_{i,j}$ are potentially large. Thus, even though $(b_{i,j} - \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle) \approx \mathbf{s}[i]\mathbf{s}[j]$, it may be the case that $h_{i,j} \cdot (b_{i,j} - \langle \mathbf{a}_{i,j}, \mathbf{t} \rangle) \not\approx h_{i,j} \cdot \mathbf{s}[i]\mathbf{s}[j]$. This is handled by considering the binary representation of $h_{i,j}$, namely $h_{i,j} = \sum_{\tau=0}^{\lfloor \log q \rfloor} 2^\tau \cdot h_{i,j,\tau}$. If, for each value of τ , we had a pair $(\mathbf{a}_{i,j,\tau}, b_{i,j,\tau})$ such that

$$b_{i,j,\tau} = \langle \mathbf{a}_{i,j,\tau}, \mathbf{t} \rangle + 2e_{i,j,\tau} + 2^\tau \mathbf{s}[i] \cdot \mathbf{s}[j] \approx \langle \mathbf{a}_{i,j,\tau}, \mathbf{t} \rangle + 2^\tau \mathbf{s}[i] \cdot \mathbf{s}[j] ,$$

then indeed

$$h_{i,j} \cdot \mathbf{s}[i]\mathbf{s}[j] = \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,j,\tau} 2^\tau \mathbf{s}[i]\mathbf{s}[j] \approx \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,j,\tau} (b_{i,j,\tau} - \langle \mathbf{a}_{i,j,\tau}, \mathbf{t} \rangle) ,$$

⁹Actually, calling these “encryptions” is inaccurate: $\mathbf{s}[i] \cdot \mathbf{s}[j] \in \mathbb{Z}_q$ is not a single bit and therefore the “ciphertext” cannot be decrypted. However, we feel that thinking of these as encryptions may benefit the reader’s intuition.

since $h_{i,j,\tau} \in \{0,1\}$. This increases the number of pairs we need to post by a factor of $(\lfloor \log q \rfloor + 1)$, which is polynomial.

This process allows us to do one multiplication without increasing the size of the ciphertext, and obtain an encryption of the product under a new secret key. *But why stop at two keys \mathbf{s} and \mathbf{t} ?* Posting a “chain” of L secret keys (together with encryptions of quadratic terms of one secret key using the next secret key) allows us to perform up to L levels of multiplications without blowing up the ciphertext size. It is possible to achieve multiplicative depth $L = \epsilon \log n$ (which corresponds to a degree $D = n^\epsilon$ polynomial) for an arbitrary constant $\epsilon < 1$ under reasonable assumptions, but beyond that, the growth of the error in the ciphertext kicks in, and destroys the ciphertext. Handling this requires us to use the machinery of bootstrapping, which we explain in the next section.

In conclusion, the above technique allows us to remove the need for “ideal assumptions” and obtain *somewhat* homomorphic encryption from LWE. This scheme will be a building block towards our full construction and is formally presented in Section 4.1.

1.2 Dimension-Modulus Reduction: Fully Homomorphic Encryption Without Squashing

As explained above, the “bootstrapping” method for achieving full homomorphism requires a \mathcal{C} -homomorphic scheme whose decryption circuit resides in \mathcal{C} . All prior somewhat homomorphic schemes fell short in this category and failed to achieve this requirement in a natural way. Thus Gentry, followed by all other previous schemes, resorted to “squashing”: a method for reducing the decryption complexity at the expense of making an additional and fairly strong assumption, namely the sparse subset sum assumption.

We show how to “upgrade” our somewhat homomorphic scheme (explained in Section 1.1) into a scheme that enjoys the same amount of homomorphism but has a much smaller decryption circuit. All of this, without making any additional assumption (beyond LWE)!

Our starting point is the somewhat homomorphic scheme from Section 1.1. Recall that a ciphertext in that scheme is of the form $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$, and decryption is done by computing $(b - \langle \mathbf{a}, \mathbf{s} \rangle \bmod q) \bmod 2$. One can verify that this computation, presented as a polynomial in the bits of \mathbf{s} , has degree at least $\max(n, \log q)$, which is more than the maximal degree D that our scheme can homomorphically evaluate. The bottom line is that decryption complexity is governed by $(n, \log q)$ which are too big for our homomorphism capabilities.

Our *dimension-modulus reduction* idea enables us to take a ciphertext with parameters $(n, \log q)$ as above, and convert it into a ciphertext of the same message, but with parameters $(k, \log p)$ which are much smaller than $(n, \log q)$. To give a hint as to the magnitude of improvement, we typically set k to be of size the security parameter and $p = \text{poly}(k)$. We can then set $n = k^c$ for essentially *any constant* c , and $q = 2^{n^\epsilon}$. We will thus be able to homomorphically evaluate functions of degree roughly $D = n^\epsilon = k^{c\epsilon}$ and we can choose c to be large enough so that this is sufficient to evaluate the $(k, \log p)$ decryption circuit.

To understand dimension-modulus reduction technically, we go back to re-linearization. We showed above that, posting proper public parameters, one can convert a ciphertext $(\mathbf{a}, b = \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m)$, that corresponds to a secret key \mathbf{s} , into a ciphertext $(\mathbf{a}', b' = \langle \mathbf{a}', \mathbf{t} \rangle + 2e' + m)$ that corresponds to a secret key \mathbf{t} .¹⁰ The crucial observation is that \mathbf{s} and \mathbf{t} need not have the same

¹⁰In the previous section, we applied re-linearization to a quadratic function of \mathbf{s} , while here we apply it to the

dimension n . Specifically, if we chose \mathbf{t} to be of dimension k , the procedure still works. This brings us down from $(n, \log q)$ to $(k, \log q)$, which is a big step but still not sufficient.

Having the above observation in mind, we wonder if we can take \mathbf{t} to have not only low dimension but also small modulus p , thus completing the transition from $(n, \log q)$ to $(k, \log p)$. This is indeed possible using some additional ideas, where the underlying intuition is that \mathbb{Z}_p can “approximate” \mathbb{Z}_q by simple scaling, up to a small error.

The public parameters for the transition from \mathbf{s} to \mathbf{t} will be $(\mathbf{a}_{i,\tau}, b_{i,\tau}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$, where

$$b_{i,\tau} = \langle \mathbf{a}_{i,\tau}, \mathbf{t} \rangle + e + \left\lfloor \frac{p}{q} \cdot 2^\tau \cdot \mathbf{s}[i] \right\rfloor. \quad .^{11}$$

Namely, we scale $2^\tau \cdot \mathbf{s}[i] \in \mathbb{Z}_q$ into an element in \mathbb{Z}_p by multiplying by p/q and rounding. The rounding incurs an additional error of magnitude at most $1/2$. It follows that

$$2^\tau \cdot \mathbf{s}[i] \approx \frac{q}{p} \cdot (b_{i,\tau} - \langle \mathbf{a}_{i,\tau}, \mathbf{t} \rangle),$$

which enables converting a linear equation in \mathbf{s} into a linear equation in \mathbf{t} . The result of dimension-modulus reduction, therefore, is a ciphertext $(\hat{\mathbf{a}}, \hat{b}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ such that $\hat{b} - \langle \hat{\mathbf{a}}, \mathbf{t} \rangle = m + 2\hat{e}$. For security, we need to assume the hardness of LWE with parameters k, p . We can show that in the parameter range we use, this assumption is as hard as the one used for the somewhat homomorphic scheme.¹²

In conclusion, dimension-modulus reduction allows us to achieve a bootstrappable scheme, based on the LWE assumption alone. We refer the reader to Section 4 for the formal presentation and full analysis of our entire solution. Specifically, dimension-modulus reduction is used for the scheme in Section 4.2.

As a nice byproduct of this technique, the ciphertexts of the resulting fully homomorphic scheme become very short! They now consist of $(k+1) \log p = O(k \log k)$ bits. This is a desirable property which is also helpful in achieving efficient private information retrieval protocols (see below).

1.3 Near-Optimal Private Information Retrieval

In (single-server) private information retrieval (PIR) protocols, a very large *database* is maintained by a *sender* (the sender is also sometimes called the server, or the database). A *receiver* wishes to obtain a specific entry in the database, without revealing any information about the entry to the server. Typically, we consider databases that are exponential in the security parameter and hence we wish that the receiver’s running time and communication complexity are polylogarithmic in the size of the database N (at least $\log N$ bits are required to specify an entry in the database). The first polylogarithmic candidate protocol was presented by Cachin, Micali and Stadler [CMS99] and additional polylogarithmic protocols were introduced by Lipmaa [Lip05] and by Gentry and Ramzan [GR05]. Of which, the latter achieves the best communication complexity

ciphertext (\mathbf{a}, b) that corresponds to a linear function of \mathbf{s} . This only makes things easier.

¹¹A subtle technical point refers to the use of an error term e , instead of $2e$ as we did for re-linearization. The reason is roughly that $\frac{q}{p} \cdot 2$ is non-integer. Therefore we “divide by 2” before performing the dimension-reduction and “multiply back” by 2 after.

¹²For the informed reader we mention that while k, p are smaller than n, q and therefore seem to imply lesser security, we are able to use much higher relative noise in our k, p scheme since it needs not support homomorphism. Hence the two assumptions are of roughly the same hardness.

of $O(\log^{3-o(1)}(N))$.¹³ The latter two protocols achieve constant amortized communication complexity when retrieving large consecutive blocks of data. See a survey in [OS07] for more details on these schemes.

Fully homomorphic, or even somewhat homomorphic, encryption is known to imply polylogarithmic PIR protocols.¹⁴ Most trivially, the receiver can encrypt the index it wants to query, and the database will use that to homomorphically evaluate the database access function, thus retrieving an encryption of the answer and sending it to the receiver. The total communication complexity of this protocol is the sum of lengths of the public key, encryption of the index and output ciphertext. However, the public key is sent only once, it is independent of the database and the query, and it can be used for many queries. Therefore it is customary to analyze such schemes in the *public key model* where sending the public key does not count towards the communication complexity. Gentry [Gen09a] proposes to use his somewhat homomorphic scheme towards this end, which requires $O(\log^3 N)$ bit communication.¹⁵ We show how, using our somewhat homomorphic scheme, in addition to new ideas, we can bring down communication complexity to a near optimal $\log N \cdot \text{polyloglog } N$ (one cannot do better than $\log N$). To obtain the best parameters, one needs to assume $2^{\tilde{\Omega}(k)}$ -hardness of polynomial-factor approximation for short vector problems in arbitrary dimension k lattices, which is supported by current knowledge. Details follow.

A major obstacle in the naive use of somewhat homomorphic encryption for PIR is that homomorphism is obtained with respect to the boolean representation of the evaluated function. Therefore, the receiver needs to encrypt the index to the database in a bit-by-bit manner. The query is then composed of $\log N$ ciphertexts, which necessitate at least $\log^2 N$ bits of communication. As a first improvement, we notice that the index needs not be encrypted under the somewhat homomorphic scheme. Rather, we can encrypt using any *symmetric* encryption scheme. The database will receive, an encrypted symmetric key (under the homomorphic scheme), which will enable it to convert symmetric ciphertexts into homomorphic ciphertexts without additional communication. The encrypted secret key can be sent as a part of the public key as it is independent of the query. This, of course, requires that our somewhat homomorphic scheme can homomorphically evaluate the decryption circuit of the symmetric scheme. Fully homomorphic schemes will certainly be adequate for this purpose, but known somewhat homomorphic schemes are also sufficient (depending on the symmetric scheme to be used). Using the most communication efficient symmetric scheme, we bring down the query complexity to $O(\log N)$. As for the sender's response, our dimension-modulus reduction technique guarantees very short ciphertexts (essentially as short as non-homomorphic LWE based schemes). This translates into $\log N \cdot \text{polyloglog } N$ bits per ciphertext, and the communication complexity of our protocol follows. We remark that in terms of retrieving large blocks of consecutive data, one can slightly reduce the overhead to $O(\log N)$ bits of communication for every bit of retrieved data. We leave it as an open problem to bring the amortized communication down to a constant. See Section 5 for the full details.

Prior to this work, it was not at all known how to achieve even polylogarithmic PIR under the LWE assumption. We stress that even if the size of the public key does count towards the

¹³It is hard to compare the performance of different PIR protocols due to the multitude of parameters. To make things easier to grasp, we compare the protocols on equal grounds: We assume that the database size and the adversary's running time are exponential in the security parameter and assume the maximal possible hardness of the underlying assumption against known attacks. We also assume that each query retrieves a single bit. We will explicitly mention special properties of individual protocols that are not captured by this comparison.

¹⁴To be precise, one needs sub-exponentially secure such schemes.

¹⁵Gentry does not provide a detailed analysis of this scheme, the above is based on our analysis of its performance.

communication complexity, our protocol still has polylogarithmic communication.

1.4 Other Related Work

Aside from Gentry’s scheme (and a variant thereof by Smart and Vercauteren [SV10] and an optimization by Stehle and Steinfeld [SS10]), there are two other fully homomorphic encryption schemes [DGHV10, BV11]. The innovation in both these schemes is the construction of a new *somewhat homomorphic* encryption scheme. Both these works then invoke Gentry’s *squashing* and bootstrapping transformation to convert it to a fully homomorphic scheme, and thus the security of both these schemes relies on the sparse subset-sum assumption (plus other assumptions). The first of these schemes is due to van Dijk, Gentry, Halevi and Vaikuntanathan [DGHV10]. Their scheme works over the integers and relies on a new assumption which, roughly speaking, states that finding the greatest common divisor of many “noisy” multiples of a number is computationally hard. They cannot, however, reduce their assumption to worst-case hardness. The second is a recent work of Brakerski and Vaikuntanathan [BV11], who construct a somewhat homomorphic encryption scheme based on the ring LWE problem [LPR10] whose security can be reduced to the worst-case hardness of problems on *ideal lattices*.

The efficiency of implementing Gentry’s scheme also gained much attention. Smart and Vercauteren [SV10], as well as Gentry and Halevi [GH11b] conduct a study on reducing the complexity of implementing the scheme.

In a recent independent work, Gentry and Halevi [GH11a] showed how the sparse subset sum assumption can be replaced by either the (decisional) Diffie-Hellman assumption or an ideal lattice assumption, by representing the decryption circuit as an arithmetic circuit with only one level of (high fan-in) multiplications.

1.5 Paper Organization

Some preliminaries and notation are described in Section 2. We formally define somewhat and fully homomorphic encryption and present the bootstrapping theorem in Section 3. The main technical section of this paper is Section 4, where our scheme is presented and fully analyzed. Lastly, our private information retrieval protocol is presented in Section 5.

2 Preliminaries

Notations. Let \mathcal{D} denote a distribution over some finite set S . Then, $x \stackrel{\$}{\leftarrow} \mathcal{D}$ is used to denote the fact that x is chosen from the distribution \mathcal{D} . When we say $x \stackrel{\$}{\leftarrow} S$, we simply mean that x is chosen from the uniform distribution over S . Unless explicitly mentioned, all logarithms are to base 2.

In this work, we utilize “noise” distributions over integers. The only property of these distributions we use is their magnitude. Hence, we define a B -bounded distribution to be a distribution over the integers where the magnitude of a sample is bounded with high probability. A definition follows.

Definition 2.1 (B -bounded distributions). *A distribution ensemble $\{\chi_n\}_{n \in \mathbb{N}}$, supported over the integers, is called B -bounded if*

$$\Pr_{e \stackrel{\$}{\leftarrow} \chi_n} [|e| > B] \leq 2^{-\tilde{\Omega}(n)} .$$

We denote scalars in plain (e.g. x) and vectors in bold lowercase (e.g. \mathbf{v}), and matrices in bold uppercase (e.g. \mathbf{A}). The ℓ_i norm of a vector is denoted by $\|\mathbf{v}\|_i$. Inner product is denoted by $\langle \mathbf{v}, \mathbf{u} \rangle$, recall that $\langle \mathbf{v}, \mathbf{u} \rangle = \mathbf{v}^T \cdot \mathbf{u}$. Let \mathbf{v} be an n dimensional vector. For all $i = 1, \dots, n$, the i^{th} element in \mathbf{v} is denoted $\mathbf{v}[i]$. We use the convention that $\mathbf{v}[0] \triangleq 1$.

We use the following variant of the leftover hash lemma [ILL89].

Lemma 2.1 (matrix-vector leftover hash lemma). *Let $\kappa \in \mathbb{N}$, $n \in \mathbb{N}$, $q \in \mathbb{N}$, and $m \geq n \log q + 2\kappa$. Let $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ be a uniformly random matrix, let $\mathbf{r} \xleftarrow{\$} \{0, 1\}^m$ and let $\mathbf{y} \xleftarrow{\$} \mathbb{Z}_q^n$. Then,*

$$\Delta((\mathbf{A}, \mathbf{A}^T \mathbf{r}), (\mathbf{A}, \mathbf{y})) \leq 2^{-\kappa}$$

where $\Delta(A, B)$ denotes the statistical distance between the distributions A and B .

2.1 Learning With Errors (LWE)

The LWE problem was introduced by Regev [Reg05] as a generalization of “learning parity with noise”. For positive integers n and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on \mathbb{Z}_q , let $A_{\mathbf{s}, \chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly at random and a noise term $e \xleftarrow{\$} \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. A formal definition follows.

Definition 2.2 (LWE). *For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over \mathbb{Z}_q , the learning with errors problem $\text{LWE}_{n,m,q,\chi}$ is defined as follows: Given m independent samples from $A_{\mathbf{s}, \chi}$ (for some $\mathbf{s} \in \mathbb{Z}_q^n$), output \mathbf{s} with noticeable probability.*

The (average-case) decision variant of the LWE problem, denoted $\text{DLWE}_{n,m,q,\chi}$, is to distinguish (with non-negligible advantage) m samples chosen according to $A_{\mathbf{s}, \chi}$ (for uniformly random $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We denote by $\text{DLWE}_{n,q,\chi}$ the variant where the adversary gets oracle access to $A_{\mathbf{s}, \chi}$, and is not a-priori bounded in the number of samples.

For cryptographic applications we are primarily interested in the average case decision problem DLWE, where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. There are known quantum [Reg05] and classical [Pei09] reductions between $\text{DLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be (discretized versions of) the Gaussian distribution, which is B -bounded for an appropriate B . Since the exact distribution χ does not matter for our results, we state a corollary of the results of [Reg05, Pei09] in terms of the bound on the distribution.

Corollary 2.2 ([Reg05, Pei09]). *Let $q = q(n) \in \mathbb{N}$ be a product of co-prime numbers $q = \prod q_i$ such that for all i , $q_i = \text{poly}(n)$, and let $B \geq n$. Then there exists an efficiently sampleable B -bounded distribution χ such that if there is an efficient algorithm that solves the (average-case) $\text{DLWE}_{n,q,\chi}$ problem. Then:*

- *There is a quantum algorithm that solves $\text{SIVP}_{\tilde{O}(n\sqrt{n} \cdot q/B)}$ and $\text{gapSVP}_{\tilde{O}(n\sqrt{n} \cdot q/B)}$ on any n -dimensional lattice, and runs in time $\text{poly}(n)$.*
- *There is a classical algorithm that solves the ζ -to- γ decisional shortest vector problem $\text{gapSVP}_{\zeta, \gamma}$, where $\gamma = \tilde{O}(n\sqrt{n} \cdot q/B)$, and $\zeta = \tilde{O}(q\sqrt{n})$, on any n -dimensional lattice, and runs in time $\text{poly}(n)$.*

We refer the reader to [Reg05, Pei09] for the formal definition of these lattice problems, as they have no direct connection to this work. We only note here that the best known algorithms for these problems run in time nearly exponential in the dimension n [AKS01, MV10]. More generally, the best algorithms that approximate these problems to within a factor of 2^k run in time $2^{\tilde{O}(n/k)}$.

2.2 Symmetric Encryption

A symmetric encryption scheme $\text{SYM} = (\text{SYM.Keygen}, \text{SYM.Enc}, \text{SYM.Dec})$, over message space $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa \in \mathbb{N}}$, is a triple of PPT algorithms as follows. We always denote the security parameter by κ .

- **Key generation.** The algorithm $sk \leftarrow \text{SYM.Keygen}(1^\kappa)$ takes a unary representation of the security parameter and outputs symmetric encryption/decryption key sk .
- **Encryption.** The algorithm $c \leftarrow \text{SYM.Enc}_{sk}(\mu)$ takes the symmetric key sk and a message $\mu \in \mathcal{M}_\kappa$ and outputs a ciphertext c .
- **Decryption.** The algorithm $\mu^* \leftarrow \text{SYM.Dec}_{sk}(c)$ takes the symmetric key sk and a ciphertext c and outputs a message $\mu^* \in \mathcal{M}_\kappa$.

Correctness and security against chosen plaintext attacks (IND-CPA security) are defined as follows.

Definition 2.3. A symmetric scheme SYM is correct if for all μ and all $sk \leftarrow \text{SYM.Keygen}(1^\kappa)$,

$$\Pr[\text{SYM.Dec}_{sk}(\text{SYM.Enc}_{sk}(\mu)) \neq \mu] = \text{negl}(\kappa) ,$$

where the probability is over the coins of SYM.Keygen , SYM.Enc .

Definition 2.4. A symmetric scheme SYM is (t, ϵ) -IND-CPA secure if for any adversary \mathcal{A} that runs in time t it holds that

$$\left| \Pr[\mathcal{A}^{\text{SYM.Enc}_{sk}(\cdot)}(1^\kappa) = 1] - \Pr[\mathcal{A}^{\text{SYM.Enc}_{sk}(0)}(1^\kappa) = 1] \right| \leq \epsilon ,$$

where the probability is over $sk \leftarrow \text{SYM.Keygen}(1^\kappa)$, the coins of SYM.Enc and the coins of the adversary \mathcal{A} .

Namely, no adversary can distinguish between an oracle that encrypts messages of its choice and an oracle that only returns encryptions of 0 (where 0 is some arbitrary element in the message space).

3 Homomorphic Encryption: Definitions and Tools

In this section we discuss the definition of homomorphic encryption and its properties as well as some related subjects. We start by defining homomorphic and fully homomorphic encryption in Section 3.1. Then, in Section 3.2 we discuss Gentry's bootstrapping theorem.

We note that there are a number of ways to define homomorphic encryption and to describe the bootstrapping theorem. We chose the definitions that best fit the constructions and we urge even the knowledgeable reader to go over them so as to avoid confusion in interpreting our results.

3.1 Homomorphic Encryption – Definitions

We now define homomorphic encryption and its desired properties. Throughout this section (and this work) we use κ to indicate the security parameter. In addition, all schemes in this paper encrypt bit-by-bit and therefore our definitions only refer to this case. The generalization to an arbitrary message space is immediate.

A homomorphic (public-key) encryption scheme $\text{HE} = (\text{HE.Keygen}, \text{HE.Enc}, \text{HE.Dec}, \text{HE.Eval})$ is a quadruple of PPT algorithms as follows.

- **Key generation.** The algorithm $(pk, evk, sk) \leftarrow \text{HE.Keygen}(1^\kappa)$ takes a unary representation of the security parameter and outputs a public encryption key pk , a public evaluation key evk and a secret decryption key sk .
- **Encryption.** The algorithm $c \leftarrow \text{HE.Enc}_{pk}(\mu)$ takes the public key pk and a single bit message $\mu \in \{0, 1\}$ and outputs a ciphertext c .
- **Decryption.** The algorithm $\mu^* \leftarrow \text{HE.Dec}_{sk}(c)$ takes the secret key sk and a ciphertext c and outputs a message $\mu^* \in \{0, 1\}$.
- **Homomorphic evaluation.** The algorithm $c_f \leftarrow \text{HE.Eval}_{evk}(f, c_1, \dots, c_\ell)$ takes the evaluation key evk , a function $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ and a set of ℓ ciphertexts c_1, \dots, c_ℓ , and outputs a ciphertext c_f .

The representation of the function f is an important issue. Since the representation can vary between schemes, we leave this issue outside of this syntactic definition. We remark, however, that in this work, f will be represented by an arithmetic circuit over $\text{GF}(2)$.

We note that while one can treat the evaluation key as a part of the public key, as has been done in the literature so far, we feel that there is an expository value to treating it as a separate entity and to distinguishing between the public elements that are used for encryption and those that are used only for homomorphic evaluation.

The only security notion we consider in this chapter is semantic security, namely security w.r.t. passive adversaries. We use its widely known formulation as IND-CPA security, defined as follows.

Definition 3.1 (CPA security). *A scheme HE is IND-CPA secure if for any polynomial time adversary \mathcal{A} it holds that*

$$\text{Adv}_{\text{CPA}}[\mathcal{A}] \triangleq |\Pr[\mathcal{A}(pk, evk, \text{HE.Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, evk, \text{HE.Enc}_{pk}(1)) = 1]| = \text{negl}(\kappa) ,$$

where $(pk, evk, sk) \leftarrow \text{HE.Keygen}(1^\kappa)$.

In fact, based on the best known about lattices, the schemes we present in this paper will be secure against even stronger adversaries. In order for our reductions to make sense for such adversaries as well, we also consider a parameterized version of CPA security. There, we allow the adversary to run in time t (which is typically super-polynomial) and succeed with probability ϵ (which is typically sub-polynomial).

Definition 3.2 $((t, \epsilon)$ -CPA security). *A scheme HE is (t, ϵ) -IND-CPA secure if for any adversary \mathcal{A} that runs in time t for $t = t(\kappa)$ it holds that*

$$\text{Adv}_{\text{CPA}}[\mathcal{A}] \triangleq |\Pr[\mathcal{A}(pk, evk, \text{HE.Enc}_{pk}(0)) = 1] - \Pr[\mathcal{A}(pk, evk, \text{HE.Enc}_{pk}(1)) = 1]| \leq \epsilon = \epsilon(\kappa) ,$$

where $(pk, evk, sk) \leftarrow \text{HE.Keygen}(1^\kappa)$.

We move on to define the homomorphism property. Note that we do not define the “correctness” of the scheme as a separate property, but rather (some form of) correctness will follow from our homomorphism properties.

We start by defining \mathcal{C} -homomorphism, which is homomorphism with respect to a specified class \mathcal{C} of functions. This notion is sometimes also referred to as “somewhat homomorphism”.

Definition 3.3 (\mathcal{C} -homomorphism). *Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ be a class of functions (together with their respective representations). A scheme HE is \mathcal{C} -homomorphic (or, homomorphic for the class \mathcal{C}) if for any sequence of functions $f_\kappa \in \mathcal{C}_\kappa$ and respective inputs $\mu_1, \dots, \mu_\ell \in \{0, 1\}$ (where $\ell = \ell(\kappa)$), it holds that*

$$\Pr[\text{HE.Dec}_{sk}(\text{HE.Eval}_{evk}(f, c_1, \dots, c_\ell)) \neq f(\mu_1, \dots, \mu_\ell)] = \text{negl}(\kappa) ,$$

where $(pk, evk, sk) \leftarrow \text{HE.Keygen}(1^\kappa)$ and $c_i \leftarrow \text{HE.Enc}_{pk}(\mu_i)$.

We point out two important properties that the above definition *does not* require. First of all, we do not require that the ciphertexts c_i are decryptable themselves, only that they become decryptable after homomorphic evaluation.¹⁶ Secondly, we do not require that the output of HE.Eval can undergo additional homomorphic evaluation.¹⁷

Before we define full homomorphism, let us define the notion of *compactness*.

Definition 3.4 (compactness). *A homomorphic scheme HE is compact if there exists a polynomial $s = s(\kappa)$ such that the output length of $\text{HE.Eval}(\dots)$ is at most s bits long (regardless of f or the number of inputs).*

Note that a \mathcal{C} -homomorphic scheme is not necessarily compact.

We give the minimal definition of fully homomorphic encryption, which suffices for most applications.

Definition 3.5 (fully homomorphic encryption). *A scheme HE is fully homomorphic if it is both compact and homomorphic for the class of all arithmetic circuits over $GF(2)$.*

As in the definition of \mathcal{C} homomorphism, one can require that the outputs of HE.Eval can again be used as inputs for homomorphic evaluation (“multi-hop homomorphism”). Indeed, all known schemes have this additional property. However, due to the complexity of the formal definition in this case, we refrain from describing a formal definition.

An important relaxation of fully homomorphic encryption is the following.

Definition 3.6 (leveled fully homomorphic encryption). *A leveled fully homomorphic encryption scheme is a homomorphic scheme where the HE.Keygen gets an additional input 1^L (now $(pk, evk, sk) \leftarrow \text{HE.Keygen}(1^\kappa, 1^L)$) and the resulting scheme is homomorphic for all depth- L binary arithmetic circuits. The bound $s(\kappa)$ on the ciphertext length must remain independent of L .*

In most cases, the only parameter of the scheme that becomes dependent on L is the bit-length of the evaluation key evk .

¹⁶Jumping ahead, while this may seem strange at first, this notion of somewhat homomorphism is all that is really required in order to bootstrap into full homomorphism and it also makes our schemes easier to describe. Lastly, note that one can always perform a “blank” homomorphic operation and then decrypt, so functionality is not hurt.

¹⁷This is termed “1-hop homomorphism” in [GHV10a].

3.2 Gentry's Bootstrapping Technique

In this section we formally define the notion of a bootstrappable encryption scheme and present Gentry's bootstrapping theorem [Gen09b, Gen09a] which implies that a bootstrappable scheme can be converted into a fully homomorphic one.

Definition 3.7 (bootstrappable encryption scheme). *Let HE be \mathcal{C} -homomorphic, and Let f_{add} and f_{mult} be the augmented decryption functions of the scheme defined as*

$$f_{\text{add}}^{c_1, c_2}(s) = \text{HE.Dec}_s(c_1) \text{ XOR } \text{HE.Dec}_s(c_2) \quad \text{and} \quad f_{\text{mult}}^{c_1, c_2}(s) = \text{HE.Dec}_s(c_1) \text{ AND } \text{HE.Dec}_s(c_2) .$$

Then \mathcal{E} is bootstrappable if

$$\{f_{\text{add}}^{c_1, c_2}, f_{\text{mult}}^{c_1, c_2}\}_{c_1, c_2} \subseteq \mathcal{C} .$$

Namely, the scheme can homomorphically evaluate f_{add} and f_{mult} .

We describe two variants of Gentry's bootstrapping theorem. The first implies leveled fully homomorphic encryption but requires no additional assumption; where the second makes an additional (*weak*) *circular security* assumption and achieves the stronger (non-leveled) variant of Definition 3.5.

The first variant follows.

Theorem 3.1 ([Gen09b, Gen09a]). *Let HE be a bootstrappable scheme, then there exists a leveled fully homomorphic encryption scheme as per Definition 3.6.*

Specifically, the leveled homomorphic scheme is such that only the length of the evaluation key depends on the level L . All other parameters of the scheme are distributed identically regardless of the value of L .

For the second variant, we need to define circular security.

Definition 3.8 (weak circular security). *A public key encryption scheme $(\text{Gen}, \text{Enc}, \text{Dec})$ is weakly circular secure if it is IND-CPA secure even for an adversary with auxiliary information containing encryptions of all secret key bits: $\{\text{Enc}_{pk}(sk[i])\}_i$.*

Namely, no polynomial time adversary can distinguish an encryption of 0 from an encryption of 1 even given the additional information.

We can now state the second theorem.

Theorem 3.2 ([Gen09b, Gen09a]). *Let HE be a bootstrappable scheme that is also weakly circular secure. Then there is a fully homomorphic encryption scheme as per Definition 3.5.*

Finally, we want to make a statement regarding the ciphertext length of a bootstrapped scheme. The following is implicit in [Gen09b, Gen09a].

Lemma 3.3. *If a scheme FH is obtained from applying either Theorem 3.1 or Theorem 3.2 to a bootstrappable scheme HE, then both FH.Enc and FH.Eval produce ciphertexts of the same length as HE.Eval (regardless of the length of the ciphertext produced by HE.Enc).*

4 The New Fully Homomorphic Encryption Scheme

In this section, we present our fully homomorphic encryption scheme and analyze its security and performance. We present our scheme in a gradual manner. First, in Section 4.1 we present an **LWE**-based somewhat homomorphic scheme, **SH**, that will serve as building block for our construction (that scheme by itself is not sufficient to achieve full homomorphism). The main technique used here is re-linearization. Our bootstrappable scheme, **BTS**, which utilizes dimension-modulus reduction, is presented in Section 4.2. We then turn to analyze the properties of **BTS**. In Section 4.3 we prove the security of the scheme based on **LWE** and discuss the worst case hardness that is implied by known reductions. In Section 4.4 we analyze the homomorphic properties of **SH** and **BTS** which enables us to prove (in Section 4.5) that the bootstrapping theorem is indeed applicable to **BTS**, and obtain a fully homomorphic scheme based on **LWE**. We then discuss the parameters and efficiency of our scheme.

4.1 The Scheme SH: A Somewhat Homomorphic Encryption Scheme

We present a somewhat homomorphic public-key encryption scheme, based on our re-linearization technique, whose message space is $\text{GF}(2)$.¹⁸ Let $\kappa \in \mathbb{N}$ be the security parameter. The scheme is parameterized by a dimension $n \in \mathbb{N}$, a positive integer $m \in \mathbb{N}$, an odd modulus $q \in \mathbb{N}$ (note that q needs not be prime) and a noise distribution χ over \mathbb{Z}_q , all of which are inherited from the **LWE** assumption we use. An additional parameter of the scheme is a number $L \in \mathbb{N}$ which is an upper bound on the maximal multiplicative depth that the scheme can homomorphically evaluate.

During the exposition of the scheme, we invite the reader to keep the following range of parameters in mind: the dimension n is polynomial in the security parameter κ , $m \geq n \log q + 2\kappa$ is a polynomial in n , the modulus is an odd number $q \in [2^{n^\epsilon}, 2 \cdot 2^{n^\epsilon})$ is sub-exponential in n (where $\epsilon \in (0, 1)$ is some constant), χ is some noise distribution that produces small samples (say, of magnitude at most n) in \mathbb{Z}_q , and the depth bound is $L \approx \epsilon \log n$.

- Key generation **SH.Keygen**(1^κ): For key generation, sample $L + 1$ vectors $\mathbf{s}_0, \dots, \mathbf{s}_L \xleftarrow{\$} \mathbb{Z}_q^n$, and compute, for all $\ell \in [L]$, $0 \leq i \leq j \leq n$, and $\tau \in \{0, \dots, \lfloor \log q \rfloor\}$, the value

$$\psi_{\ell,i,j,\tau} := \left(\mathbf{a}_{\ell,i,j,\tau}, b_{\ell,i,j,\tau} := \langle \mathbf{a}_{\ell,i,j,\tau}, \mathbf{s}_\ell \rangle + 2 \cdot e_{\ell,i,j,\tau} + 2^\tau \cdot \mathbf{s}_{\ell-1}[i] \cdot \mathbf{s}_{\ell-1}[j] \right) \in \mathbb{Z}_q^n \times \mathbb{Z}_q, \quad (1)$$

where $\mathbf{a}_{\ell,i,j,\tau} \xleftarrow{\$} \mathbb{Z}_q^n$, $e_{\ell,i,j,\tau} \xleftarrow{\$} \chi$ (recall that, according to our notational convention, $\mathbf{s}_{\ell-1}[0] \triangleq 1$). We define $\Psi \triangleq \{\psi_{\ell,i,j,\tau}\}_{\ell,i,j,\tau}$ to be the set of all these values.¹⁹ At this point, it may not yet be clear what the purpose of the 2^τ factors is; indeed, this will be explained later when we explain homomorphic multiplication.

The key-generation algorithm proceeds to choose a uniformly random matrix $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and a vector $\mathbf{e} \xleftarrow{\$} \chi^m$, and compute $\mathbf{b} := \mathbf{A}\mathbf{s}_0 + 2\mathbf{e}$.

It then outputs the secret key $sk = \mathbf{s}_L$, the evaluation key $evk = \Psi$, and the public key $pk = (\mathbf{A}, \mathbf{b})$.²⁰

¹⁸It is quite straightforward to generalize the scheme to work over a message space $\text{GF}(t)$, where t is relatively prime to q . Since we mostly care about the binary case, we choose not to present this generalization.

¹⁹A knowledgeable reader may notice that the above is similar to encryptions of $2^\tau \cdot \mathbf{s}_{\ell-1}[i] \cdot \mathbf{s}_{\ell-1}[j] \pmod{q}$ via an **LWE**-based scheme, except this “ciphertext” is not decryptable since the “message” is not a single bit value.

²⁰The public key pk is essentially identical to the public key in Regev’s scheme.

- Encryption $\text{SH.Enc}_{pk}(\mu)$: Recall that $pk = (\mathbf{A}, \mathbf{b})$. To encrypt a message $\mu \in \text{GF}(2)$, sample a vector $\mathbf{r} \xleftarrow{\$} \{0, 1\}^m$ and set (just like in Regev's scheme)

$$\mathbf{v} := \mathbf{A}^T \mathbf{r} \quad \text{and} \quad w := \mathbf{b}^T \mathbf{r} + \mu .$$

The output ciphertext contains the pair (\mathbf{v}, w) , in addition to a “level tag” which is used during homomorphic evaluation and indicates the “multiplicative depth” where the ciphertext has been generating. For freshly encrypted ciphertext, therefore, the level tag is zero. Formally, the encryption algorithm outputs $c := ((\mathbf{v}, w), 0)$.

- Homomorphic evaluation $\text{SH.Eval}_{evk}(f, c_1, \dots, c_t)$ where $f : \{0, 1\}^t \rightarrow \{0, 1\}$: We require that f is represented by a binary arithmetic circuit with ‘+’ gates of arbitrary fan-in and ‘×’ gates with fan-in 2. We further require that the circuit is *layered*, namely that it is composed of homogenous layers of either all ‘+’ gates or all ‘×’ gates (it is easy to see that any arithmetic circuit can be converted to this form). Lastly, we require that the multiplicative depth of the circuit (the total number of ‘×’ layers) is exactly L .²¹

We homomorphically evaluate the circuit f gate by gate. Namely, we will show how to perform homomorphic addition (of arbitrarily many ciphertexts) and homomorphic multiplication (of two ciphertexts). Combining the two, we will be able to evaluate any such function f .

Ciphertext structure during evaluation. During the homomorphic evaluation, we will generate ciphertexts of the form $c = ((\mathbf{v}, w), \ell)$, where the tag ℓ indicates the multiplicative level at which the ciphertext has been generated (hence fresh ciphertexts are tagged with 0). The requirement that f is layered will make sure that throughout the homomorphic evaluation all inputs to a gate have the same tag. In addition, we will keep the invariant that the output of each gate evaluation $c = ((\mathbf{v}, w), \ell)$, is such that

$$w - \langle \mathbf{v}, \mathbf{s}_\ell \rangle = \mu + 2 \cdot e \pmod{q} , \quad (2)$$

where μ is the correct plaintext output of the gate, and e is a noise term that depends on the gate's input ciphertexts. Note that it always holds that $\ell \leq L$ due to the bound on the multiplicative depth, and that the output of the homomorphic evaluation of the entire circuit is expected to have $\ell = L$.

Homomorphic evaluation of gates:

- *Addition gates.* Homomorphic evaluation of a ‘+’ gate on inputs c_1, \dots, c_t , where $c_i = ((\mathbf{v}_i, w_i), \ell)$, is performed by outputting

$$c_{\text{add}} = ((\mathbf{v}_{\text{add}}, w_{\text{add}}), \ell) := \left(\left(\sum_i \mathbf{v}_i, \sum_i w_i \right), \ell \right) .$$

Informally, one can see that

$$w_{\text{add}} - \langle \mathbf{v}_{\text{add}}, \mathbf{s}_\ell \rangle = \sum_i (w_i - \langle \mathbf{v}_i, \mathbf{s}_\ell \rangle) = \sum_i (\mu_i + 2e_i) = \sum_i \mu_i + 2 \sum_i e_i ,$$

²¹Jumping ahead, in the analysis we will only prove correctness for a specific sub-class of these circuits.

where μ_i is the plaintext corresponding to μ_i . The output of the homomorphic evaluation, thus, corresponds to the sum of the inputs, with the noise term being the sum of input noises.

- *Multiplication gates.* We show how to multiply ciphertexts c, c' where $c = ((\mathbf{v}, w), \ell)$ and $c' = ((\mathbf{v}', w'), \ell)$ (recall that multiplication gates have fan-in 2), to obtain an output ciphertext $c_{\text{mult}} = ((\mathbf{v}_{\text{mult}}, w_{\text{mult}}), \ell + 1)$. Note that the level tag increases by 1.

We first consider an n -variate *symbolic* polynomial over the unknown vector \mathbf{x} :

$$\phi(\mathbf{x}) = \phi_{(w, \mathbf{v}), (w', \mathbf{v}')}(\mathbf{x}) \triangleq (w - \langle \mathbf{v}, \mathbf{x} \rangle) \cdot (w' - \langle \mathbf{v}', \mathbf{x} \rangle) . \quad (3)$$

We symbolically open the parenthesis of this quadratic polynomial, and express it as

$$\phi(\mathbf{x}) = \sum_{0 \leq i \leq j \leq n} h_{i,j} \cdot \mathbf{x}[i] \cdot \mathbf{x}[j] ,$$

where $h_{i,j} \in \mathbb{Z}_q$ are known (we can compute them from $(\mathbf{v}, w), (\mathbf{v}', w')$ by opening parenthesis in Eq. (3)).²²

For technical reasons (related to keeping the error growth under control), we want to express $\phi(\cdot)$ as a polynomial with small coefficients. We consider the binary representation of $h_{i,j}$, letting $h_{i,j,\tau}$ be the τ^{th} bit in this representation. In other words

$$h_{i,j} = \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,j,\tau} \cdot 2^\tau ,$$

for $h_{i,j,\tau} \in \{0, 1\}$.

We can express ϕ therefore as

$$\phi(\mathbf{x}) = \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot (2^\tau \cdot \mathbf{x}[i] \cdot \mathbf{x}[j]) .^{23}$$

We recall that the evaluation key $evk = \Psi$ contains elements of the form $\psi_{\ell,i,j,\tau} = (\mathbf{a}_{\ell,i,j,\tau}, b_{\ell,i,j,\tau})$ such that

$$2^\tau \mathbf{s}_\ell[i] \mathbf{s}_\ell[j] \approx b_{\ell+1,i,j,\tau} - \langle \mathbf{a}_{\ell+1,i,j,\tau}, \mathbf{s}_{\ell+1} \rangle .$$

The homomorphic multiplication algorithm will thus set

$$\mathbf{v}_{\text{mult}} := \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot \mathbf{a}_{\ell+1,i,j,\tau} ,$$

and

$$w_{\text{mult}} = \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot b_{\ell+1,i,j,\tau} ,$$

²²We once again remind the reader that because of the notational trick of setting $\mathbf{x}[0] \triangleq 1$, this expression captures the constant term in the product, as well as all the linear terms, thus homogenizing the polynomial $\phi(\mathbf{x})$.

²³This can be interpreted as a polynomial with small coefficients whose variables are $(2^\tau \cdot \mathbf{x}[i] \cdot \mathbf{x}[j])$.

The final output ciphertext will be

$$c_{\text{mult}} := ((\mathbf{v}_{\text{mult}}, w_{\text{mult}}), \ell + 1) .$$

Note that the level tag is increased by one as expected. Let us now verify that our invariant as per Eq. 2 still holds for the new ciphertext:

$$\begin{aligned}
w_{\text{mult}} - \langle \mathbf{v}_{\text{mult}}, \mathbf{s}_{\ell+1} \rangle &= \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot (b_{\ell+1,i,j,\tau} - \langle \mathbf{a}_{\ell+1,i,j,\tau}, \mathbf{s}_{\ell+1} \rangle) \\
&= \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot 2^\tau \cdot \mathbf{s}_\ell[i] \cdot \mathbf{s}_\ell[j] + 2 \cdot h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \\
&= \phi(\mathbf{s}_\ell) + \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \\
&= (w - \langle \mathbf{v}, \mathbf{s}_\ell \rangle) \cdot (w' - \langle \mathbf{v}', \mathbf{s}_\ell \rangle) + \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \\
&= (\mu + 2e)(\mu' + 2e') + \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} 2 \cdot h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \\
&= \mu\mu' + 2 \left(\mu e' + \mu' e + 2ee' + \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \right) . \quad (4)
\end{aligned}$$

Indeed, we get the plaintext output $\mu\mu'$ in addition to a noise term that is inherited from the input ciphertexts and from the evaluation key.

- Decryption $\text{SH.Dec}_{\mathbf{s}_L}(c)$: To decrypt a ciphertext $c = ((\mathbf{v}, w), L)$ (recall that we are only required to decrypt ciphertexts that are output by $\text{SH.Eval}(\dots)$ and those will always have level tag L), compute

$$(w - \langle \mathbf{v}, \mathbf{s}_L \rangle \pmod{q}) \pmod{2} . \quad (5)$$

4.2 The Scheme BTS: A Bootstrappable Scheme

We now utilize the dimension-modulus reduction technique to present the scheme BTS, which uses SH as building block and inherits its homomorphic properties. However, BTS has much shorter ciphertexts and lower decryption complexity, which will enable us to apply the bootstrapping theorem to obtain full homomorphism.

Our bootstrappable scheme is parameterized by (n, m, q, χ, L) , which are the parameters for SH, and additional parameters $(k, p, \hat{\chi})$ which are the “smaller” parameters. $n, q \in \mathbb{N}$ are referred to as the “long” dimension and modulus respectively, while k, p are the “short” dimension and modulus. $\chi, \hat{\chi}$ are the long and short noise distributions, over \mathbb{Z}_q and \mathbb{Z}_p , respectively. The parameter $m \in \mathbb{N}$ is used towards public key generation. The parameter L is an upper bound on the multiplicative depth of the evaluated function.

While we discuss parameter values below, we encourage the reader to consider the following (non-optimal, but easier to understand) settings as a running example: $k = \kappa$, $n = k^4$, $q \approx 2^{\sqrt{n}}$, $L = 1/3 \log n = 4/3 \log k$, $p = (n^2 \log q) \cdot \text{poly}(k) = \text{poly}(k)$, $m = O(n \log q)$. The distributions $\chi, \hat{\chi}$ can be thought of as being n - and k -bounded, respectively.

- Key generation $\text{BTS.Keygen}(1^\kappa)$: Run $\text{SH.Keygen}(1^\kappa)$ to obtain the secret key \mathbf{s}_L , evaluation key Ψ and public key (\mathbf{A}, \mathbf{b}) of SH .

Recall that $\mathbf{s}_L \in \mathbb{Z}_q^n$, $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}_q^{m \times n} \times \mathbb{Z}_q^m$, and $\Psi \in (\mathbb{Z}_q^n \times \mathbb{Z}_q)^{(n+1)^2 \cdot (\lceil \log q \rceil + 1) \cdot L}$.

Proceed by sampling the “short” secret key $\hat{\mathbf{s}} \xleftarrow{\$} \mathbb{Z}_p^k$ and computing additional parameters for the evaluation key: For all $i \in [n]$, $\tau \in \{0, \dots, \lceil \log q \rceil\}$, sample $\hat{\mathbf{a}}_{i,\tau} \xleftarrow{\$} \mathbb{Z}_p^k$, $\hat{e}_{i,\tau} \xleftarrow{\$} \hat{\chi}$, and compute

$$\hat{b}_{i,\tau} := \langle \hat{\mathbf{a}}_{i,\tau}, \hat{\mathbf{s}} \rangle + \hat{e}_{i,\tau} + \left\lfloor \frac{p}{q} \cdot (2^\tau \cdot \mathbf{s}_L[i]) \right\rfloor \pmod{p}.$$

Set $\hat{\psi}_{i,\tau} := (\hat{\mathbf{a}}_{i,\tau}, \hat{b}_{i,\tau}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$, and

$$\hat{\Psi} := \{\hat{\psi}_{i,\tau}\}_{i \in [n], \tau \in \{0, \dots, \lceil \log q \rceil\}}.$$

This is very similar to the generation of Ψ in the scheme SH , but now $\hat{\psi}_{i,\tau}$ “encodes” scaled linear terms, rather than quadratic terms.

Finally, output the secret key $sk = \hat{\mathbf{s}}$, evaluation key $evk = (\Psi, \hat{\Psi})$ and public key $pk = (\mathbf{A}, \mathbf{b})$. Note that the public key is identical to that of SH .

- Encryption $\text{BTS.Enc}_{pk}(\mu)$: Use the same encryption algorithm as SH . To encrypt a bit $\mu \in \{0, 1\}$, compute $c \leftarrow \text{SH.Enc}_{(\mathbf{A}, \mathbf{b})}(\mu)$ and output c as the ciphertext.
- Homomorphic evaluation $\text{BTS.Eval}_{evk}(f, c_1, \dots, c_t)$, where $f : \{0, 1\}^t \rightarrow \{0, 1\}$: Recall that $evk = (\Psi, \hat{\Psi})$. To perform homomorphic evaluation, we will use the homomorphic evaluation function of SH . We thus require that f is represented by a binary arithmetic circuit which is a legal input for SH.Eval .

The first step in the homomorphic evaluation is computing

$$c_f \leftarrow \text{SH.Eval}_\Psi(f, c_1, \dots, c_t).$$

This results in a ciphertext of the form $c_f = ((\mathbf{v}, w), L) \in \mathbb{Z}_q^n \times \mathbb{Z}_q \times \{L\}$.

Next, we reduce the dimension and modulus of c_f to k, p as follows. Consider the following function from \mathbb{Z}^n into the rationals modulo p

$$\phi(\mathbf{x}) \triangleq \phi_{\mathbf{v}, w}(\mathbf{x}) \triangleq \frac{p}{q} \cdot \left(\frac{q+1}{2} \cdot (w - \langle \mathbf{v}, \mathbf{x} \rangle) \right) \pmod{p}.$$

Rearranging, one can find $h_0, \dots, h_n \in \mathbb{Z}_q$ such that

$$\phi(\mathbf{x}) = \sum_{i=0}^n h_i \cdot \left(\frac{p}{q} \cdot \mathbf{x}[i] \right) \pmod{p},$$

Let $h_{i,\tau}$ be the τ^{th} bit of h_i , for all $\tau \in \{0, \dots, \lfloor \log q \rfloor\}$. Then

$$\phi(\mathbf{x}) = \sum_{i=0}^n \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \left(\frac{p}{q} \cdot 2^\tau \cdot \mathbf{x}[i] \right) .$$

Using the parameters in $\hat{\Psi}$, we create a new ciphertext $\hat{c} = (\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ by setting

$$\begin{aligned} \hat{\mathbf{v}} &:= 2 \cdot \sum_{i=0}^n \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \hat{\mathbf{a}}_{i,\tau} \pmod{p} \in \mathbb{Z}_p^k \\ \hat{w} &:= 2 \cdot \sum_{i=0}^n \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \hat{b}_{i,\tau} \pmod{p} \in \mathbb{Z}_p . \end{aligned}$$

The output of **BTS.Eval** is the new ciphertext $\hat{c} \in \mathbb{Z}_p^k \times \mathbb{Z}_p$. Note that the bit-length of \hat{c} is $(k+1) \log p$.

Recall the invariant we enforce on the structure of ciphertexts of **SH** (see Eq. 2). We show that a similar invariant holds for \hat{c} : Namely, that if c_f is such that $w - \langle \mathbf{v}, \mathbf{s}_L \rangle = \mu + 2e \pmod{q}$, then

$$\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle = \mu + 2\hat{e} \pmod{p} ,$$

where \hat{e} is proportional to $\frac{p}{q}e$ (an appropriately scaled version of e) plus some additional noise.

To see the above, recall that $(p+1)/2$ is the inverse of 2 modulo p , and notice that²⁴

$$\begin{aligned} \frac{p+1}{2} (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) &= \sum_{i=0}^n \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \cdot \left(\hat{b}_{i,\tau} - \langle \hat{\mathbf{a}}_{i,\tau}, \hat{\mathbf{s}} \rangle \right) \pmod{p} \\ &= \sum_{i=0}^n \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} \left(\hat{c}_{i,\tau} + \left\lfloor \frac{p}{q} \cdot (2^\tau \cdot \mathbf{s}_L[i]) \right\rfloor \right) \pmod{p} \\ &= \phi(\mathbf{s}_L) + \underbrace{\sum_{i=0}^n \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} (\hat{c}_{i,\tau} + \hat{\omega}_{i,\tau})}_{\triangleq \delta_1} \pmod{p} , \end{aligned} \tag{6}$$

where we define

$$\hat{\omega}_{i,\tau} \triangleq \left\lfloor \frac{p}{q} \cdot (2^\tau \cdot \mathbf{s}_L[i]) \right\rfloor - \frac{p}{q} \cdot (2^\tau \cdot \mathbf{s}_L[i]) ,$$

and notice that $|\hat{\omega}_{i,\tau}| \leq 1/2$. Since $h_{i,\tau} \in \{0, 1\}$ and $\hat{c}_{i,\tau}$ is small, δ_1 (defined in Eq. (6)) is “small” as well.

²⁴While the following sequence of derivations might seem like an indirect way to prove what we need, the way we choose to do it will be useful later.

Now, letting $w = \langle \mathbf{v}, \mathbf{s}_L \rangle + 2e + \mu \pmod{q}$, we wish to examine $\phi(\mathbf{s}_L) \triangleq \phi_{\langle \mathbf{v}, w \rangle}(\mathbf{s}_L)$ more closely, as follows.

$$\begin{aligned}
\phi(\mathbf{s}_L) &\triangleq \frac{p}{q} \cdot \left(\frac{q+1}{2} \cdot (w - \langle \mathbf{v}, \mathbf{s}_L \rangle) \right) \pmod{p} \\
&= \frac{p}{q} \cdot \left(\frac{q+1}{2} \cdot (2e + \mu + Mq) \right) \pmod{p} \quad (\text{where } M \in \mathbb{Z}) \\
&= \frac{p}{q} \cdot \left(\frac{q+1}{2} \mu + e + M'q \right) \pmod{p} \quad (\text{where } M' = M + e \in \mathbb{Z}) \\
&= \frac{p}{q} \cdot \frac{q+1}{2} \mu + \frac{p}{q} \cdot e \pmod{p} \\
&= \frac{p+1}{2} \cdot \mu + \underbrace{\left(\frac{p}{q} - 1 \right) \cdot \frac{\mu}{2} + \frac{p}{q} \cdot e}_{\triangleq \delta_2} \pmod{p} \\
&= \frac{p+1}{2} \cdot \mu + \delta_2
\end{aligned} \tag{7}$$

and notice that if $p \leq q$ (as is the case in our setting), $|\delta_2| \leq \frac{p}{q} |e| + \frac{1}{2}$.

Putting together Eq. (6) and (7), we see that

$$\frac{p+1}{2} (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) = \frac{p+1}{2} \cdot \mu + (\delta_1 + \delta_2) . \tag{8}$$

Multiplying by 2, we have

$$\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle = \mu + 2(\delta_1 + \delta_2) . \tag{9}$$

Now defining $\hat{e} \triangleq \delta_1 + \delta_2$, the invariant follows.

It is important to notice that, while not immediate from its definition, $\hat{e} = \delta_1 + \delta_2$ is an integer. To see this, note that it can be represented as a difference between integers:

$$\delta_1 + \delta_2 = \frac{p+1}{2} (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle) - \frac{p+1}{2} \cdot \mu .$$

- Decryption $\text{BTS.Dec}_{\hat{\mathbf{s}}}(\hat{c})$: To decrypt $\hat{c} = (\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ (recall, again, that we only need to decrypt ciphertexts that are output by BTS.Eval), compute

$$\mu^* := (\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle \pmod{p}) \pmod{2} .$$

If indeed $\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle = \mu + 2\hat{e} \pmod{p}$ then $\mu^* = \mu$ so long as \hat{e} is small enough.

4.3 Security Analysis

In this section, we analyze the security of **BTS** based on **LWE** and then, using known connections, based on worst case hardness of lattice problems.

The following theorem asserts the security of **BTS** based on two **DLWE** problems: One with modulus q , dimension n and noise χ , and one with modulus p , dimension k and noise $\hat{\chi}$.

Theorem 4.1 (security). *Let $n = n(\kappa), k = k(\kappa), q = q(\kappa), p = p(\kappa)$ and $L = L(\kappa)$ be functions of the security parameter. Let $\chi, \hat{\chi}$ be some distributions over the integers, and define $m \triangleq n \log q + 2\kappa$.*

The scheme BTS is CPA secure under the $\text{DLWE}_{n,q,\chi}$ and the $\text{DLWE}_{k,p,\hat{\chi}}$ assumptions. In particular, if both the $\text{DLWE}_{n,q,\chi}$ and the $\text{DLWE}_{k,p,\hat{\chi}}$ problems are (t, ϵ) -hard, then the scheme is $(t - \text{poly}(\kappa), 2(L+1) \cdot (2^{-\kappa} + \epsilon))$ -semantically secure.

Essentially, the view of a CPA adversary for our scheme is very similar to Regev's scheme, with the exception that our adversary also gets to see the evaluation key. However, the evaluation key contains a sequence of LWE instances which, based on our assumption, are indistinguishable from uniform. Therefore our reduction will perform a sequence of L hybrids to replace the Ψ component of the evaluation key with a set of completely uniform elements. Then, an additional hybrid will imply the same for $\hat{\Psi}$. Once this is done, we will use the known proof techniques from Regev's scheme and get the security of our scheme. A formal proof follows.

Proof. As explained above, we prove by a sequence of hybrids. Let \mathcal{A} be an IND-CPA adversary for BTS that runs in time t . We consider a series of hybrids where $\text{Adv}_H[\mathcal{A}]$ denotes the success probability of \mathcal{A} in hybrid H .

- **Hybrid \hat{H}_{L+1} :** This is identical to the IND-CPA game, where the adversary gets properly distributed keys pk, evk , generated by BTS.Keygen , and an encryption of either 0 or 1 computed using BTS.Enc . By definition,

$$\text{Adv}_{\hat{H}_{L+1}}[\mathcal{A}] \triangleq |\Pr[\mathcal{A}(pk, \text{SH.Enc}_{pk}(\mu_0)) = 1] - \Pr[\mathcal{A}(pk, \text{SH.Enc}_{pk}(\mu_1)) = 1]| = \delta.$$

- **Hybrid H_{L+1} :** This hybrid is identical to \hat{H}_{L+1} in everything except the generation of $\hat{\Psi}$. In this hybrid, $\hat{\Psi}$ is not generated as prescribed, but is rather sampled uniformly. Namely, for all i, τ we set $\hat{\psi}_{i,\tau} \xleftarrow{\$} \mathbb{Z}_p^k \times \mathbb{Z}_p$.

It follows that there exists an adversary $\hat{\mathcal{B}}$ that solves the $\text{DLWE}_{k,p,\hat{\chi}}$ problem in time $t + \text{poly}(\kappa)$ and advantage

$$\text{DLWE}_{k,p,\hat{\chi}} \text{Adv}[\hat{\mathcal{B}}] \geq 1/2 \cdot |\text{Adv}_{\hat{H}_{L+1}}[\mathcal{A}] - \text{Adv}_{H_{L+1}}[\mathcal{A}]|.$$

The adversary $\hat{\mathcal{B}}$ will sample all vectors $\mathbf{s}_0, \dots, \mathbf{s}_L$ by himself and generate pk, Ψ . Then, he will use the LWE oracle to obtain either $A_{\hat{\mathbf{s}}, \hat{\chi}}$ samples which will result in properly generated $\hat{\Psi}$, or uniform samples which will result in a uniform $\hat{\Psi}$. $\hat{\mathcal{B}}$ will then sample a uniform $b \xleftarrow{\$} \{0, 1\}$ and return 1 if and only if $\mathcal{A}(pk, (\Psi, \hat{\Psi}), \text{BTS.Enc}_{pk}(b)) = b$. Using simple algebra, the result follows.

- **Hybrid H_ℓ , for $\ell \in [L]$:** Hybrid H_ℓ is identical to $H_{\ell+1}$, except for a change in the Ψ component of the evaluation key. Specifically, we change each of the components $\psi_{\ell,i,j,\tau}$ for all i, j, τ : Instead of computing $\psi_{\ell,i,j,\tau}$ as prescribed (i.e., $(\mathbf{a}_{\ell,i,j,\tau}, \langle \mathbf{a}_{\ell,i,j,\tau}, \mathbf{s}_\ell \rangle + 2e_{\ell,i,j,\tau} + 2^\tau \cdot \mathbf{s}_{\ell-1}[i] \cdot \mathbf{s}_{\ell-1}[j])$), we sample it uniformly. Namely, we set $\psi_{\ell,i,j,\tau} \xleftarrow{\$} \mathbb{Z}_q^n \times \mathbb{Z}_q$.

It follows that there exists an adversary \mathcal{B}_ℓ that solves the $\text{DLWE}_{n,q,\chi}$ problem in time $t + \text{poly}(\kappa)$ and advantage

$$\text{DLWE}_{n,q,\chi} \text{Adv}[\mathcal{B}_\ell] = 1/2 \cdot |\text{Adv}_{H_\ell}[\mathcal{A}] - \text{Adv}_{H_{\ell+1}}[\mathcal{A}]|.$$

The argument is very similar to the previous hybrid: We note that at this point $\hat{\Psi}$ and $\{\psi_{\lambda,i,j,\tau}\}_{\lambda>\ell,i,j,\tau}$ are completely uniform and can be generated without any knowledge of $\mathbf{s}_{\ell+1}, \dots, \mathbf{s}_L, \hat{\mathbf{s}}$. The adversary \mathcal{B}_ℓ will sample all vectors $\mathbf{s}_0, \dots, \mathbf{s}_{\ell-1}$ himself, and turn to the LWE oracle for samples in order to generate $\psi_{\ell,i,j,\tau}$. This will result in Ψ being identical to $H_{\ell+1}$ if the oracle returns $A_{\mathbf{s},\chi}$ samples, or Ψ being identical to H_ℓ if the oracle returns uniform elements. Once again, sampling a random b and checking whether \mathcal{A} 's response is identical to \mathcal{B} completes the argument.

Note that in the hybrid H_1 , the evaluation key $evk = (\Psi, \hat{\Psi})$ is completely uniform, and hence the view of the adversary is like in Regev's scheme.

- **Hybrid H_0 :** Hybrid H_0 is identical to H_1 except that the vector \mathbf{b} in the public key is chosen uniformly at random from \mathbb{Z}_q^m , rather than being computed as $\mathbf{A} \cdot \mathbf{s}_0 + 2\mathbf{e}$. Under the $\text{DLWE}_{n,q,\chi}$ assumption, hybrids H_0 and H_1 are indistinguishable. Namely, there exists an adversary \mathcal{B}_0 that runs in time $t + \text{poly}(\kappa)$ and whose advantage is

$$\text{DLWE}_{n,q,\chi} \text{Adv}[\mathcal{B}_0] = 1/2 \cdot |\text{Adv}_{H_1}[\mathcal{A}] - \text{Adv}_{H_0}[\mathcal{A}]| .$$

The adversary \mathcal{B}_0 gets m samples from the LWE oracle and uses them to generate (\mathbf{A}, \mathbf{b}) . If the samples come from $A_{\mathbf{s},\chi}$, then \mathbf{b} is distributed like in H_1 and if they are uniform then \mathbf{b} is distributed as in H_0 . The same testing of \mathcal{A} as before implies the argument.

- **Hybrid H_{rand} :** Hybrid H_{rand} is identical to H_0 except that the ciphertext is chosen uniformly at random from $\mathbb{Z}_q^n \times \mathbb{Z}_q$, rather than being computed as $(\mathbf{A}^T \cdot \mathbf{r}, \mathbf{b}^T \cdot \mathbf{r} + \mu)$.

We now claim that

$$|\text{Adv}_{H_0}[\mathcal{A}] - \text{Adv}_{H_{\text{rand}}}[\mathcal{A}]| \leq 2^{-\kappa} .$$

This is due to the Leftover hash lemma (Lemma 2.1), since $m > (n+1) \log q + 2\kappa$.

Note that in H_{rand} , all the elements of both the public key and the ciphertext are uniformly random and independent of the message. Thus,

$$\text{Adv}_{H_{\text{rand}}}[\mathcal{A}] = 0 .$$

Putting these together, we get that

$$\text{Adv}_{\text{CPA}}[\mathcal{A}] \leq 2^{-\kappa} + 2 \cdot \left(\text{DLWE}_{k,p,\hat{\chi}} \text{Adv}[\hat{\mathcal{B}}] + \sum_{\ell=0}^L \text{DLWE}_{n,q,\chi} \text{Adv}[\mathcal{B}_\ell] \right) ,$$

and the result follows. \square

Specific Parameters and Worst-Case Hardness. The parameters we require for homomorphism (see Theorem 4.2 below) are as follows. We require that $q = 2^{n^\epsilon}$ for some $\epsilon \in (0, 1)$, χ is n -bounded, $p = 16nk \log(2q)$ and $\hat{\chi}$ is k -bounded. In order to achieve the best lattice reduction, we will choose q as a product of polynomially bounded co-prime numbers. Applying known results (see Corollary 2.2), $\text{DLWE}_{n,q,\chi}$ translates into approximating short-vector problems in worst case

n -dimensional lattices to within a factor of $\tilde{O}(\sqrt{n} \cdot 2^{n^\epsilon})$, while $\text{DLWE}_{k,p,\chi}$ translates to approximating k -dimensional lattice problems to within $\tilde{O}(n^{1+\epsilon} \cdot k^{1.5})$ factor.²⁵ These problems are essentially incomparable as the hardness of the problem increases as the dimension increases on one hand, but decreases as the approximation factor increases on the other. The best known algorithms solve the first problem in time (roughly) $2^{\tilde{O}(n^{1-\epsilon})}$, and the second in time $2^{\tilde{O}(k)}$.

The relation between n and k is determined based on the required homomorphic properties. In this work, we only prove there exists a constant C such that setting $n = k^{C/\epsilon}$ implies fully homomorphic encryption. Given the value of C , setting $\epsilon \approx 1 - \frac{1}{C+1}$ will make the two problems equally hard (at least based on the current state of the art).

4.4 Homomorphic Properties of SH And BTS

In this section we analyze the homomorphic properties of SH and BTS. Both schemes have essentially the same homomorphic properties but BTS has the additional advantage of having low decryption complexity (as analyzed in Section 4.5). Thus, BTS would be our main focus, and the properties of SH will follow as a by-product of our analysis.

We start by formally defining the class of functions for which we prove homomorphism and proceed by stating the homomorphic properties and proving them.

The Function Class $\text{Arith}[L, T]$. In this section we define the function class for which we prove somewhat homomorphism of our scheme. Essentially, this is the class of arithmetic circuits over $\text{GF}(2)$ with bounded fan-in and bounded depth, with an additional final “collation”: a high fan-in addition gate at the last level. We require that the circuit is structured in a canonical “layered” manner as we describe below.

Definition 4.1. *Let $L = L(\kappa), T = T(\kappa)$ be functions of the security parameter. The class $\text{Arith}[L, T]$ is the class of arithmetic circuits over $\text{GF}(2)$, with $\{+, \times\}$ gates, with the following structure. Each circuit contains exactly $2L + 1$ layers of gates (numbered $1, \dots, 2L + 1$ starting from the input level), gates of layer $i + 1$ are fed only by gates of layer i . The odd layers contain only ‘+’ gates and the even layers contain only ‘ \times ’ gates. The gates at layers $1, \dots, 2L$ have fan-in 2, while the final addition gate in layer $2L + 1$ is allowed to have fan-in T .*

We note that $\text{Arith}[L, T]$ conforms with the requirements on the evaluated function imposed by SH.Eval and BTS.Eval . Indeed, the multiplicative depth of any circuit in $\text{Arith}[L, T]$ is exactly L , and hence, homomorphic evaluation is well defined on any such function.

To motivate the choice of this function class, we first note that any arithmetic circuit of fan-in 2 and depth D can be trivially converted into a circuit in $\text{Arith}[D, 1]$.²⁶ This will be useful for the purpose of bootstrapping. Jumping ahead, the collation gate will be useful for constructing a private information retrieval protocol, where we will need to evaluate polynomials with a very large number of monomials and fairly low degree. The collation gate will thus be used for the final aggregation of monomials.

²⁵We do not mention the specific lattice problem or the specific type of reduction (quantum vs. classical) since, as one can observe from Corollary 2.2, the approximation factor we get is essentially the same for all problems, and the state of the art is roughly the same as well.

²⁶One way to do this is to separate each level of the circuit into two levels – an addition level and a multiplication level – and finally, adding a dummy fan-in-1 addition gate at the top. This gives us a $2D + 1$ depth circuit with alternating addition and multiplication levels, or, in other words, the transformed circuit belongs to $\text{Arith}[D, 1]$.

Our goal is now to prove that with the appropriate choice of parameters, SH and BTS are $\text{Arith}[L, T]$ -homomorphic.

Theorem 4.2. *Let $n = n(\kappa) \geq 5$ be any polynomial, $q \geq 2^{n^\epsilon} \geq 3$ for some $\epsilon \in (0, 1)$ be odd, χ be any n -bounded distribution, and $m = (n + 1) \log q + 2\kappa$. Let $k = \kappa$, $p = 16nk \log(2q)$ (odd) and $\hat{\chi}$ be any k -bounded distribution. Then SH and BTS are both $\text{Arith}[L = \Omega(\epsilon \log n), T = \sqrt{q}]$ -homomorphic.*

Not surprisingly, the homomorphism class depends only on n and not on k . This is because, recalling the definition of BTS.Eval , the homomorphism property is inherited from SH.Eval . We note that it is possible to further generalize the class of circuits that we can homomorphically evaluate (for example, circuits with high multiplicative depth but low multiplicative degree), however since this is not required for our results, and since the proof will use the exact same tools, we choose not to further complicate the theorem statement and proof.

To prove the theorem, we introduce a sequence of lemmas as follows. Recall that the encryption algorithms of both schemes are identical, and that BTS.Eval first calls SH.Eval on all its inputs. We first analyze the growth of the noise in the execution of SH.Eval in Lemma 4.3 (which will imply the theorem for SH), and then, in Lemma 4.4, we complete the noise calculation of BTS.Eval , which will complete the proof of the theorem.

To track the growth of the noise, we define, for any ciphertext $c = ((\mathbf{v}, w), \ell)$ a noise measure $\eta(c) \in \mathbb{Z}$ as follows. We let $e \in \mathbb{Z}$ be the smallest integer (in absolute value) such that

$$\mu + 2e = w - \langle \mathbf{v}, \mathbf{s}_d \rangle \pmod{q},$$

and define $\eta(c) \triangleq \mu + 2e$ (note that $\eta(c)$ is defined over the integers, and not modulo q). We note that so long as $|\eta(c)| < q/2$, the ciphertext is decryptable. We can now bound the error in the execution by bounding $\eta(c_f)$ of the output ciphertext.

Lemma 4.3. *Let $n = n(\kappa) \geq 5$, $q = q(\kappa) \geq 3$, χ be B -bounded and $L = L(\kappa)$ and let $f \in \text{Arith}[L, T]$, $f : \{0, 1\}^t \rightarrow \{0, 1\}$ (for some $t = t(\kappa)$). Then for any input $\mu_1, \dots, \mu_t \in \{0, 1\}$, if we let $(pk, evk, sk) \leftarrow \text{SH.Keygen}(1^\kappa)$, $c_i \leftarrow \text{BTS.Enc}_{pk}(\mu_i) = \text{SH.Enc}_{pk}(\mu_i)$ and we further let $c_f = ((\mathbf{v}, w), L) \leftarrow \text{SH.Eval}_{evk}(f, c_1, \dots, c_t)$ be the encryption of $f(\mu_1, \dots, \mu_t)$, it holds that with all but negligible probability*

$$|\eta(c_f)| \leq T \cdot (16nB \log q)^{2^L}.$$

Proof. We assume that all samples of χ (there are only polynomially many of them) are indeed of magnitude at most B . This happens with all but exponentially small probability. The remainder of the analysis is completely deterministic.

We track the growth of noise as the homomorphic evaluation proceeds.

- **Fresh ciphertexts.** Our starting point is level-0 ciphertexts $((\mathbf{v}, w), 0)$ that are generated by the encryption algorithm. By definition of the encryption algorithm we have that

$$w - \langle \mathbf{v}, \mathbf{s}_0 \rangle = \mathbf{r}^T \cdot \mathbf{b} + \mu - \mathbf{r}^T \cdot \mathbf{A} \cdot \mathbf{s}_0 = \mu + \mathbf{r}^T \cdot (\mathbf{b} - \mathbf{A} \mathbf{s}_0) = \mu + 2\mathbf{r}^T \cdot \mathbf{e} \pmod{q}.$$

Since $|\mu + 2\mathbf{r}^T \cdot \mathbf{e}| \leq 1 + 2nB$, it follows that

$$|\eta(c)| \leq 2nB + 1. \tag{10}$$

- **Homomorphic addition gates.** When evaluating ‘+’ on ciphertexts c_1, \dots, c_t to obtain c_{add} , we just sum their (\mathbf{v}, w) values. Therefore

$$|\eta(c_{\text{add}})| \leq \sum_i |\eta(c_i)| .$$

- **Homomorphic multiplication gates.** When evaluating ‘ \times ’ on $c = ((\mathbf{v}, w), \ell)$, $c' = ((\mathbf{v}', w'), \ell)$ to obtain $c_{\text{mult}} = ((\mathbf{v}_{\text{mult}}, w_{\text{mult}}), \ell + 1)$, we get that by Eq. (4)

$$w_{\text{mult}} - \langle \mathbf{v}_{\text{mult}}, \mathbf{s}_{\ell+1} \rangle = \eta(c) \cdot \eta(c') + 2 \sum_{\substack{0 \leq i \leq j \leq n \\ \tau \in \{0, \dots, \lfloor \log q \rfloor\}}} h_{i,j,\tau} \cdot e_{\ell+1,i,j,\tau} \pmod{q} .$$

It follows that

$$|\eta(c_{\text{mult}})| \leq |\eta(c)| \cdot |\eta(c')| + 2 \cdot \frac{(n+1)(n+2)}{2} \cdot B(\log q + 1) .$$

If we define

$$E \triangleq \max \left\{ |\eta(c)|, |\eta(c')|, (n+2)\sqrt{B \log(2q)} \right\} ,$$

then $|\eta(c_{\text{mult}})| \leq 2E^2$.

Let

$$E_0 = \max \left\{ 2nB + 1, (n+2)\sqrt{B \log(2q)} \right\} \leq 2nB \log q$$

be an upper bound on $|\eta(c)|$ of fresh ciphertexts.

Then it holds that a bound $E_{2\ell}$ on $|\eta(c)|$ of the outputs of layer 2ℓ (recall that the even layers contain multiplication gates) is obtained by

$$E_{2\ell} \leq 2(2E_{2(\ell-1)})^2 .$$

and therefore, recursively,

$$E_{2L} \leq (8E_0)^{2^L} \leq (16nB \log q)^{2^L} .$$

And after the final collation gate it holds that

$$|\eta(c_f)| \leq T \cdot (16nB \log q)^{2^L} . \quad \square$$

We now similarly define $\hat{\eta}(\hat{c})$ for $\hat{c} = (\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$ that encrypts μ . We let $\hat{e} \in \mathbb{Z}$ be the smallest integer (in absolute value) such that

$$\mu + 2\hat{e} = \hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle \pmod{p} ,$$

and define $\hat{\eta}(\hat{c}) \triangleq \mu + 2\hat{e}$ (note that, as before, $\hat{\eta}$ is defined over the integers, and not modulo p). So long as $|\hat{\eta}(\hat{c})| < p/2$, **BTS.Dec** will decrypt \hat{c} correctly. In the next lemma, we bound $|\hat{\eta}(\hat{c})|$ of the output of **BTS.Eval**.

Lemma 4.4. *Let $n = n(\kappa) \geq 5$, $q = q(\kappa) \geq 3$, χ be B -bounded and $L = L(\kappa)$. Let $p = p(\kappa)$, $k = k(\kappa)$ and $\hat{\chi}$ be \hat{B} -bounded. Consider a homomorphic evaluation $\hat{c} \leftarrow \text{BTS.Eval}_{\text{evk}}(f, c_1, \dots, c_t)$ and the terms δ_1, δ_2 defined in Eq. (6) and (7), respectively. Let $c_f \in \mathbb{Z}_q^n \times \mathbb{Z}_q \times \{L\}$ be the intermediate value returned by the call to SH.Eval . Then with all but negligible probability*

$$|\delta_1 + \delta_2| \leq \frac{p}{2q} |\eta(c_f)| + 2n\hat{B} \log(2q) .$$

Proof. We assume that all samples from $\hat{\chi}$ are indeed of magnitude at most \hat{B} . This happens with all but exponentially small probability.

By definition (recall that δ_1, δ_2 have been defined over the rationals), we have that

$$|\delta_1| = \left| \sum_{i=0}^n \sum_{\tau=0}^{\lfloor \log q \rfloor} h_{i,\tau} (\hat{e}_{i,\tau} + \hat{\omega}_{i,\tau}) \right| \leq (n+1) \log(2q) (\hat{B} + 1/2) ,$$

and

$$\begin{aligned} |\delta_2| &= \left| \left(\frac{p}{q} - 1 \right) \cdot \frac{\mu}{2} + \frac{p}{q} \cdot e \right| \\ &= \left| \frac{p}{q} \cdot \frac{\mu + 2e}{2} - \frac{\mu}{2} \right| \\ &\leq \frac{p}{2q} |\eta(c_f)| + 1/2 . \end{aligned}$$

Adding the terms together, the result follows. \square

We can now finally prove Theorem 4.2.

Proof of Theorem 4.2. Let us consider the homomorphism claim about **BTS** (the argument for **SH** will follow as by-product): A sufficient condition for ciphertext $\hat{c} = (\hat{\mathbf{v}}, \hat{w})$ to decrypt correctly is that $\hat{e} < p/4$. By Lemma 4.4, it is sufficient to prove that

$$p/4 > \frac{p}{2q} |\eta(c_f)| + 2n\hat{B} \log(2q) \geq \frac{p}{2q} |\eta(c_f)| + p/8 .$$

Thus it is sufficient to prove that

$$|\eta(c_f)| < q/4 .$$

We note that if we prove this, then it also follows that c_f is decryptable and hence the claim about the homomorphism of **SH** holds as well.

Plugging in the bound from Lemma 4.3, we get

$$T \cdot (16nB \log q)^{2^L} < q/4 ,$$

and plugging in all the parameters and $T = \sqrt{q}$, we need

$$(16n^{2+\epsilon})^{2^L} < 2^{n^\epsilon/2}/4$$

which clearly holds for some $L = \Omega(\epsilon \log n)$. \square

4.5 Bootstrapping and Full Homomorphism

We now show how to apply Gentry's bootstrapping theorem (Theorems 3.1, 3.2) to achieve full homomorphism. In order to do this, we first need to bound the complexity of an augmented decryption circuit. Since our decryption is essentially a computation of inner product, we bound the complexity of this operation.

Lemma 4.5. *Let $(\hat{\mathbf{v}}, \hat{w}) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$. There exists an arithmetic circuit with fan-in 2 gates and $O(\log k + \log \log p)$ depth, that on input $\hat{\mathbf{s}} \in \mathbb{Z}_p^k$ (in binary representation) computes*

$$(\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle \pmod{p}) \pmod{2}.$$

Proof. We let $\hat{\mathbf{s}}[i](j)$ denote the j^{th} bit of the binary representation of $\hat{\mathbf{s}}[i] \in \mathbb{Z}_p$. We notice that

$$\begin{aligned} \hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle &= \hat{w} - \sum_{i=1}^k \hat{\mathbf{s}}[i] \hat{\mathbf{v}}[i] \pmod{p} \\ &= \hat{w} - \sum_{i=1}^k \sum_{j=0}^{\lfloor \log p \rfloor} \hat{\mathbf{s}}[i](j) \cdot (2^j \cdot \hat{\mathbf{v}}[i]) \pmod{p}. \end{aligned}$$

Therefore computing $\hat{w} - \langle \hat{\mathbf{v}}, \hat{\mathbf{s}} \rangle \pmod{p}$ is equivalent to summing up $k(1 + \lfloor \log p \rfloor) + 1$ numbers in \mathbb{Z}_p , and then taking the result modulo p . The summation (over the integers) can be done in depth $O(\log k + \log \log p)$. In order to take modulo p , one needs to subtract, in parallel, all possible multiples of p (there are at most $O(k \log p)$ options) and check if the result is in \mathbb{Z}_p . This requires depth $O(\log k + \log \log p)$ again. Then a selection tree of depth $O(\log k + \log \log p)$ is used to choose the correct result. Once this is done, outputting the least significant bit implements the final modulo 2 operation.

The total depth is thus $O(\log k + \log \log p)$ as required. \square

We can now apply the bootstrapping theorem to obtain a fully homomorphic scheme.

Lemma 4.6. *There exists $C \in \mathbb{N}$ such that setting $n = k^{C/\epsilon}$ and the rest of the parameters as in Theorem 4.2, BTS is bootstrappable as per Definition 3.7.*

Proof. Lemma 4.5 guarantees that the decryption circuit is in $\text{Arith}[O(\log k), 1]$ (note that $\log \log p = o(\log k)$), since the augmented decryption circuit just adds 1 to the depth, it follows that the augmented decryption circuits are also in $\text{Arith}[O(\log k), 1]$.

Theorem 4.2, on the other hand, guarantees homomorphism for any $\text{Arith}[\Omega(\epsilon \log n), \sqrt{q}]$ function. Taking a large enough C , it will hold that $\text{Arith}[O(\log k), 1] \subseteq \text{Arith}[\Omega(\epsilon \log n), \sqrt{q}]$ and the lemma follows. \square

Finally, we conclude that there exists an LWE based fully homomorphic encryption based on Theorem 4.1 and Lemma 4.6.

Corollary 4.7. *There exists a leveled fully homomorphic encryption based on the $\text{DLWE}_{n,q,\chi}$ and $\text{DLWE}_{k,p,\hat{\chi}}$ assumptions.*

Furthermore, if BTS is weakly circular secure (see Definition 3.8), then there exists a fully homomorphic encryption based on the same assumptions.

Efficiency of the Scheme. Interestingly, our scheme is comparable to *non-homomorphic* LWE based schemes (e.g. Regev’s) in terms of encryption, decryption and ciphertext sizes. Namely, so long as one doesn’t use the homomorphic properties of the scheme, she does not need to “pay” for it. To see why this is the case, we observe that our scheme’s secret key has length $k \log p = O(\kappa \log \kappa)$ and the ciphertext length is $(k + 1) \log p = O(\kappa \log \kappa)$. The decryption algorithm is essentially the same as Regev’s. As far as encryption is concerned, it may seem more costly. The public key as we describe it contains $(n + 1)((n + 1) \log q + 2\kappa) \log q$ bits, and encryption requires performing operations over \mathbb{Z}_q . However, we note that one can think of sampling a public key $(\hat{\mathbf{A}}, \hat{\mathbf{b}})$ where $\hat{\mathbf{A}} \xleftarrow{\$} \mathbb{Z}_p^{m \times k}$, $\hat{\mathbf{b}} = \hat{\mathbf{A}}\hat{\mathbf{s}} + 2\hat{\mathbf{e}} \in \mathbb{Z}_p^m$ (where $m = ((k + 1) \log p + 2\kappa)$). This will enable generating short ciphertexts that will be “bootstrapped up” during the homomorphic evaluation. If such short public key is used, then encryption also becomes comparable to Regev’s scheme.

Homomorphic evaluation is where the high price is paid. the evaluation key has size $O(Ln^2 \log^2 q + n \log q \log p) = \tilde{O}(n^{2+2\epsilon})$. Considering the fact that $n = \kappa^{C/\epsilon}$, this accumulates to a fairly long evaluation key, especially considering that in a leveled scheme, this size increases linearly with the depth of the circuit to be evaluated. The bright side, as we mention above, is that *evk* only needs to be known to the homomorphic evaluator and is not needed for encryption or decryption.

Circuit Privacy. A property that is sometimes desired in the context of fully homomorphic encryption is *circuit privacy*. A scheme is circuit private if the output of a homomorphic evaluation, reveals no information on the evaluated function (other than the output of the function on the encrypted message). Circuit privacy for our scheme can be achieved by adding additional noise to the ciphertext c_f , right before applying dimension-modulus reduction. Similar techniques were used in previous schemes and thus we feel that a more elaborate discussion is unnecessary here.

5 LWE-Based Private Information Retrieval

In this section, we present a single-server private information retrieval (PIR) protocol with nearly optimal communication complexity. First, we present the definitions of PIR in Section 5.1. Then, in Section 5.2, we show a generic construction of PIR from somewhat homomorphic encryption. Finally, in Section 5.3, we instantiate the generic construction using our own scheme from Section 4 and analyze its parameters.

5.1 Definitions of Single Server PIR

We define single server private information retrieval in the public-key setting. In this setting, there is a public key associated with the receiver (who holds the respective secret key). This public key is independent of the query and of the database, and can be generated and sent (or posted) before the interaction begins, and may be used many times. Thus, the size of the public key is not counted towards communication complexity of the scheme. We formalize this by an efficient setup procedure that runs before the protocol starts and generate this public key.

Letting κ be the security parameter and let $N \in \mathbb{N}$ be the database size, a PIR protocol PIR in the public-key setting is defined by a tuple of polynomial-time computable algorithms (PIR.Setup, PIR.Query, PIR.Response, PIR.Decode) as follows:

0. **Setup.** The protocol begins in an off-line setup phase that does not depend on the index to be queried nor on the contents of the database.

The receiver runs the setup algorithm

$$(params, setupstate) \leftarrow \text{PIR.Setup}(1^\kappa) .$$

It thus obtains a public set of parameters $params$ (the public key) that is sent to the sender, and a secret state $setupstate$ that is kept private.

Once the setup phase is complete, the receiver and sender can run the remainder of the protocol an unbounded number of times.

1. **Query.** When the receiver wishes to receive the i^{th} element in the database $\text{DB}[i]$, it runs

$$(query, qstate) \leftarrow \text{PIR.Query}(1^\kappa, setupstate, i) .$$

The query message $query$ is then sent to the sender and $qstate$ is a query-specific secret information that is kept private.

2. **Answer.** The sender has access to a database $\text{DB} \in \{0,1\}^N$. Upon receiving the query message $query$ from the receiver, it runs the “answering” algorithm

$$resp \leftarrow \text{PIR.Response}(1^\kappa, \text{DB}, params, query) .$$

The response $resp$ is then sent back to the receiver.

3. **Decode.** Upon receiving $resp$, the receiver decodes the response by running

$$x \leftarrow \text{PIR.Decode}(1^\kappa, setupstate, qstate, resp) .$$

The output $x \in \{0,1\}$ is the output of the protocol.

We note that while in general a multi-round interactive protocol is required for each database query, the protocols we present are of the simple form of a query message followed by a response message. Hence, we chose to present the simple syntax above.

The communication complexity of the protocol is defined to be $|query| + |resp|$. Namely, the number of bits being exchanged to transfer a single database element (excluding the setup phase). We sometime analyze the query length and the response length separately.

Correctness and security are defined as follows.

- **Correctness.** For all $\kappa \in \mathbb{N}$, $\text{DB} \in \{0,1\}^*$ where $N \triangleq |\text{DB}|$, and $i \in [N]$, it holds that

$$\Pr[\text{PIR.Decode}(1^\kappa, setupstate, qstate, resp) \neq \text{DB}[i]] = \text{negl}(\kappa) ,$$

where $(params, setupstate) \leftarrow \text{PIR.Setup}(1^\kappa)$, $(query, qstate) \leftarrow \text{PIR.Query}(1^\kappa, setupstate, i)$ and $resp \leftarrow \text{PIR.Response}(1^\kappa, \text{DB}, params, query)$.

- **(t, ϵ)-Privacy.** For all $\kappa \in \mathbb{N}$, $N \in \mathbb{N}$ and for any adversary \mathcal{A} running in time $t = t_{\kappa, N}$ it holds that

$$\max_{\substack{\mathbf{i}=(i_1, \dots, i_t), \\ \mathbf{j}=(j_1, \dots, j_t) \in [N]^t}} |\Pr[\mathcal{A}(params, \mathbf{i}, query_{\mathbf{i}}) = 1] - \Pr[\mathcal{A}(params, \mathbf{j}, query_{\mathbf{j}}) = 1]| \leq \epsilon \quad (= \epsilon_{\kappa, N}) ,$$

where $(params, setupstate) \leftarrow \text{PIR.Setup}(1^\kappa)$, $(query_{i_\ell}, qstate_{i_\ell}) \leftarrow \text{PIR.Query}(1^\kappa, setupstate, i_\ell)$ and $(query_{j_\ell}, qstate_{j_\ell}) \leftarrow \text{PIR.Query}(1^\kappa, setupstate, j_\ell)$, for all $\ell \in [t]$.

We note that the definition of privacy above differs from the one usually found in literature. The standard definition refers to vectors \mathbf{i}, \mathbf{j} of dimension 1. That is, only allow the adversary to see one query to the database. A hybrid argument can show that with proper degradation in parameters, this guarantees some security also for the case of many queries. However in the public-key setting, where the same public key is used for all queries, this hybrid argument no longer works. Thus, we must require that the adversary is allowed to view many query strings.²⁷ In fact, one could consider even stronger attacks in the public-key setting, which is outside the scope of this work.

The definition of privacy deserves some further discussion. We note that we did not define the ranges of parameters for (t, ϵ) for which the protocol is considered “private”. Indeed there are several meaningful ways to define what it means for a protocol to be private. Let us discuss two options and provide corresponding definitions.

- i. The first approach is to argue that the resources of the adversary are similar to those of an honest server (we can think of an adversary as a “server gone bad”). Thus, in this approach the adversary can run in polynomial time in N, κ and must still not succeed with non-negligible probability in N, κ . We say that a scheme is (i) -private if it is $(p(\kappa, N), 1/p(\kappa, N))$ -private for any polynomial $p(\cdot, \cdot)$.
- ii. The second approach argues that the security parameter is the “real” measure for privacy. Thus the protocol needs to be exponentially secure in the security parameter. Thus a scheme is (ii) -private if it is $(2^{\Omega(\kappa)}, 2^{-\Omega(\kappa)})$ -private.

5.2 PIR via Somewhat Homomorphic and Symmetric Encryption

In this section we describe a generic PIR protocol that uses a somewhat homomorphic encryption and an arbitrary symmetric encryption as building blocks. This protocol has the useful property that the somewhat homomorphic scheme is not used to encrypt the index to the database. Rather, we use the symmetric scheme to encrypt the index, and have the server homomorphically decrypt it during query evaluation. Thus, the receiver’s query can be rather short.

Our PIR protocol relies on two building blocks – a semantically secure symmetric encryption scheme $\text{SYM} = (\text{SYM.Keygen}, \text{SYM.Enc}, \text{SYM.Dec})$ over the message space $[N]$, and a somewhat homomorphic encryption scheme $\text{HE} = (\text{HE.Keygen}, \text{HE.Enc}, \text{HE.Dec}, \text{HE.Eval})$. The level of somewhat homomorphism required for the protocol depends on the symmetric scheme being used (in particular, the decryption complexity of the symmetric scheme). We recall that in Section 4, we get a leveled fully homomorphic scheme without relying on any circular security assumptions, this means that it can be used together with any symmetric scheme. However, a clever selection of the symmetric scheme to be used can make our methodology applicable also for somewhat homomorphic schemes, such as the scheme BTS from Section 4, even without bootstrapping.

We present a protocol $\text{PIR} = (\text{PIR.Setup}, \text{PIR.Query}, \text{PIR.Response}, \text{PIR.Decode})$ (as defined in Section 5.1).

- $\text{PIR.Setup}(1^\kappa)$: In the setup procedure, we generate a symmetric key $\text{symsk} \leftarrow \text{SYM.Keygen}(1^\kappa)$ and keys for the somewhat homomorphic scheme $(\text{hpk}, \text{hevk}, \text{hsk}) \leftarrow \text{HE.Keygen}(1^\kappa)$.

²⁷We feel that our definition captures the essence of an attack on a PIR protocol more than the standard one-time definition, even in the usual setting. As we mention above, converting between the definitions incurs a linear blowup in the adversary’s advantage so a (t, ϵ) -private scheme according to the old definition is only $(t, t\epsilon)$ -private according to ours.

The symmetric key is then encrypted using the homomorphic public key to create a ciphertext

$$c_{symk} \leftarrow \text{HE.Enc}_{hpk}(symk) .$$

We note that if HE is bit encryption scheme, then $symk$ is encrypted bit by bit.

The setup procedure then outputs the public parameters

$$params := (hevk, c_{symk}) ,$$

and the secret state

$$setupstate := (hpk, hsk, symk) .$$

- $\text{PIR.Query}(1^\kappa, setupstate, i)$: To generate a query string, we just encrypt i using the symmetric scheme. Recall that $setupstate = (hpk, hsk, symk)$, then

$$query \leftarrow \text{SYM.Enc}_{symk}(i) .$$

In our scheme, no additional information needs to be saved per query: $qstate := \phi$.

- $\text{PIR.Response}(1^\kappa, \text{DB}, params, query)$: Upon receiving a query, a response is computed as follows. Recall that $params = (hevk, c_{symk})$ and consider the function h defined as follows:

$$h(x) \triangleq \text{DB}[\text{SYM.Dec}(x, query)] ,$$

namely the function h uses its input as a symmetric key to decrypt the query, and then uses the plaintext to index the database and retrieve the appropriate value. Note that $h(symk) = \text{DB}[i]$, where i is the index embedded in $query$.

While PIR.Response does not know $symk$, it does know c_{symk} and thus can homomorphically evaluate $h(symk)$ and set

$$resp \leftarrow \text{HE.Eval}_{hevk}(h, c_{symk}) .$$

Note that $resp$ should correspond to a decryptable ciphertext of $\text{DB}[i]$.

- $\text{PIR.Decode}(1^\kappa, setupstate, qstate, resp)$: We recall that $setupstate = (hpk, hsk, symk)$ and that $qstate$ is null. To decode the answer to the query, we decrypt the ciphertext associated with $resp$, outputting

$$b \leftarrow \text{HE.Dec}_{hsk}(resp) .$$

Correctness and privacy are easily reduced to those of the underlying primitives in the following lemmas.

Lemma 5.1 (correctness). *If our symmetric scheme SYM, and our somewhat homomorphic scheme HE are correct and if the somewhat homomorphic scheme can evaluate the function h defined above, then our PIR protocol is correct.*

Proof. Since HE is correct with regards to homomorphic evaluation, then with all but negligible probability $b = h(symk)$. Since SYM is correct, it follows that $h(symk) = \text{DB}[i]$ with all but negligible probability. \square

Lemma 5.2 (privacy). *If our somewhat homomorphic scheme is $(t \cdot \text{poly}(\kappa), \epsilon_1)$ -CPA secure and our symmetric scheme is $(t + \text{poly}(\kappa), \epsilon_2)$ -CPA secure, then our PIR protocol is $(t, 2(\epsilon_1 + \epsilon_2))$ -private.*

Proof. We prove this by a series of hybrids (or experiments). Let \mathcal{A} be an adversary that runs in time t against the privacy of our protocol and has advantage ϵ . We consider the behavior of \mathcal{A} in a number of hybrids H_0, H_1, H_2 as defined below. We let $\text{Adv}_{H_i}[\mathcal{A}]$ denote the advantage of \mathcal{A} in hybrid H_i .

- **Hybrid H_0 .** This is identical to the original privacy game of the scheme. By definition

$$\text{Adv}_{H_0}[\mathcal{A}] = \epsilon .$$

- **Hybrid H_1 .** We now change the game so that instead of computing $c_{\text{sym}sk} \leftarrow \text{HE.Enc}_{hp}(sym_{sk})$ in PIR.Setup , we will set $c_{\text{sym}sk} \leftarrow \text{HE.Enc}_{hp}(0)$.

There exists an adversary \mathcal{B} for the CPA-security of the somewhat homomorphic scheme that runs in time $t \cdot \text{poly}(\kappa)$ and whose advantage is

$$\text{CPAAdv}[\mathcal{B}] = (1/2) \cdot |\text{Adv}_{H_0}[\mathcal{A}] - \text{Adv}_{H_1}[\mathcal{A}]| .$$

It follows that

$$|\text{Adv}_{H_0}[\mathcal{A}] - \text{Adv}_{H_1}[\mathcal{A}]| \leq 2\epsilon_1 .$$

- **Hybrid H_2 .** We now change the game so that instead of setting $query_\ell \leftarrow \text{SYM.Enc}_{sym_{sk}}(i_\ell)$ in PIR.Query , we will set $query_\ell \leftarrow \text{SYM.Enc}_{sym_{sk}}(0)$ for all $\ell \in [1 \dots t]$.

There exists an adversary \mathcal{C} for the CPA-security of the symmetric scheme that runs in time $t + \text{poly}(\kappa)$ and whose advantage is

$$\text{CPAAdv}[\mathcal{C}] = (1/2) \cdot |\text{Adv}_{H_1}[\mathcal{A}] - \text{Adv}_{H_2}[\mathcal{A}]| .$$

It follows that

$$|\text{Adv}_{H_1}[\mathcal{A}] - \text{Adv}_{H_2}[\mathcal{A}]| \leq 2\epsilon_2 .$$

However, in H_2 , the view of the adversary is independent of the queried indices. Therefore

$$\text{Adv}_{H_2}[\mathcal{A}] = 0 .$$

It follows that $\epsilon \leq 2(\epsilon_1 + \epsilon_2)$ as required. \square

Lastly, let us analyze the communication complexity of our protocol. It follows by definition that the query size is the length of an encryption of $\{0, 1\}^{\lceil \log N \rceil}$ bits using our symmetric scheme, and the response is the encryption of a single bit using our somewhat homomorphic scheme.

5.3 Instantiating the Components: The PIR Protocol

We show how to implement the primitives required in Section 5.2 in two different ways.

An Explicit LWE-Based Solution. The first idea is to use an optimized, symmetric-key LWE-based encryption as the symmetric encryption scheme in the PIR protocol, together with our scheme BTS as the homomorphic scheme. Specifically, using the same parameters k, p as in our bootstrappable scheme, we get a symmetric scheme whose decryption is almost identical to that of our bootstrappable scheme.

In particular, we apply an optimization of [PVW08, ACPS09] to get ciphertexts of length $O(\log N) + O(k \log k)$ to encrypt $\log N$ bits of the index. Roughly speaking, the optimization is based on two observations: first, rather than encrypting a single bit using an element of \mathbb{Z}_p , we can “pack in” $O(\log p)$ bits, if we set the error in the LWE instances to be correspondingly smaller (but still a $1/\text{poly}(k)$ fraction of p). Secondly, observe that in a symmetric ciphertext $(\mathbf{v}, w) \in \mathbb{Z}_p^k \times \mathbb{Z}_p$, most of the space is consumed by the vector \mathbf{v} . The observation of [PVW08, ACPS09] is that \mathbf{v} can be re-used to encrypt multiple messages using different secret keys $\mathbf{s}_1, \dots, \mathbf{s}_\ell$. Using these optimizations, the resulting PIR protocol has query length of $O(k \log k + \log N)$ bits and response length $O(k \log k)$ for $k = \text{poly}(\kappa)$. The following corollary summarizes the properties of this scheme.

Corollary 5.3 ([PVW08, ACPS09]). *Let $p, k, \hat{\chi}$ be as in Theorem 4.2. Then there exists a $\text{DLWE}_{k,p,\hat{\chi}}$ -secure symmetric encryption scheme whose ciphertext length is $O(k \log k + \ell)$ for ℓ -bit messages, and whose decryption circuit has the same depth as that of BTS.Dec .*

Recall the analysis of BTS from Section 4. We can prove that the function h can be evaluated homomorphically.

Lemma 5.4. *Let SYM be the scheme from Corollary 5.3, then $h(x) \triangleq \text{DB}[\text{SYM.Dec}_x(\text{query})]$ is such that $h \in \text{Arith}[O(\log k) + \log \log N, N]$.*

Proof. We implement h as follows. First, we decrypt the value of query to obtain an index i . Then, we compute the function $\sum_{j \in [N]} \text{DB}[j] \cdot \mathbb{1}_{i=j}$. The decryption circuit is implemented in depth $O(\log k)$ as in Lemma 4.5. The function $\mathbb{1}_{i=j}$ is implemented using a comparison tree of depth $\log \log N$. Finally, a collation gate of fan-in N is used to compute the final sum. The result follows. \square

This means that we can choose n to be large enough such that h can be evaluated by BTS.

Theorem 5.5. *There exists a PIR protocol with communication complexity $O(k \log k + \log N)$ based on the $\text{DLWE}_{n,q,\chi}$ and $\text{DLWE}_{k,p,\hat{\chi}}$ assumptions, for $n = \text{poly}(k)$ and the remainder of the parameters as in Theorem 4.2.*

Proof. We choose n such that $L = \Omega(\epsilon \log n) > O(\log k) + \log \log N$ and such that $\sqrt{q} = 2^{n^\epsilon/2} \geq N$. This will result in $n = \text{poly}(k, \log N)$ (recall that the communication complexity depends only on k). The result follows from Theorem 4.2 and Theorem 4.3. \square

For the best currently known attacks on LWE (see [MR09, LP11, RS10]), this protocol is $(2^{\Omega(k/\text{polylog } k)}, 2^{-\Omega(k/\text{polylog } k)})$ -private. Thus, going back to our definitions in Section 5.1, and setting $k = \kappa \cdot \text{polylog}(\kappa)$, we get a (ii) -private PIR scheme with a total communication complexity of $O(\log N) + O(\kappa \cdot \text{polylog}(\kappa))$; and a (i) -private scheme with communication complexity $\log N \cdot \text{polyloglog}(N)$ by setting $\kappa = \log N \cdot \text{polyloglog}(N) = \omega(\log N)$.

An Almost Optimal Solution Using Pseudorandom Functions. A second instantiation aims to bring the (ii) -private communication complexity down to $\log N + \kappa \cdot \text{polylog}(\kappa)$. This can be done by instantiating the symmetric encryption scheme above with an optimal symmetric encryption scheme with ciphertexts of length $\log N + \kappa \cdot \text{polylog}(\kappa)$. Such a scheme follows immediately given any pseudo-random function (PRF).

If we want to base security solely on **LWE**, we can use the **LWE**-based PRF that is obtained by applying the GGM transformation [GGM86] to an **LWE** based pseudorandom generator. Note that using such instantiation, we cannot argue that $h \in \text{Arith}[L, T]$ for reasonable L, T (since the complexity of evaluating the PRF might be high). However, we can use our leveled fully homomorphic scheme to support the required circuit depth of any function, and in particular the aforementioned PRF.

The Complexity of Transmitting the Public Parameters. Finally, we note that the parameters produced in the setup phase of our protocol are of length $\text{poly}(\kappa)$. Thus our protocol can be trivially modified to work in a setting without setup, with communication complexity $\log N + \text{poly}(\kappa)$ (under the (ii) -private notion) and $\text{polylog}(N)$ (under the (i) -private notion).

References

- [ABB10] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.
- [ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.
- [AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.
- [Ajt98] Miklós Ajtai. The shortest vector problem in \mathcal{L}_2 is p -hard for randomized reductions (extended abstract). In *STOC*, pages 10–19, 1998.
- [AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.
- [BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Theory of Cryptography - TCC'05*, volume 3378 of *Lecture Notes in Computer Science*, pages 325–341. Springer, 2005.
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, 2011. To appear.
- [CHKP10] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

- [CMS99] Christian Cachin, Silvio Micali, and Markus Stadler. Computationally private information retrieval with polylogarithmic communication. In *EUROCRYPT*, pages 402–414, 1999.
- [DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010. Full Version in <http://eprint.iacr.org/2009/616.pdf>.
- [Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.
- [Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [Gen10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In *CRYPTO*, pages 116–137, 2010.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.
- [GH11a] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. Cryptology ePrint Archive, Report 2011/279, 2011. <http://eprint.iacr.org/2011/279>.
- [GH11b] Craig Gentry and Shai Halevi. Implementing gentry’s fully-homomorphic encryption scheme. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.
- [GHV10a] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. *i*-hop homomorphic encryption and rerandomizable Yao circuits. In *CRYPTO*, pages 155–172, 2010.
- [GHV10b] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, pages 506–522, 2010.
- [GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377. ACM, 1982.
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC*, pages 197–206. ACM, 2008.
- [GR05] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *ICALP*, volume 3580 of *Lecture Notes in Computer Science*, pages 803–815. Springer, 2005.
- [ILL89] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions (extended abstracts). In *STOC*, pages 12–24. ACM, 1989.

- [IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594. Springer, 2007.
- [Lip05] Helger Lipmaa. An oblivious transfer protocol with log-squared communication. In Jianying Zhou, Javier Lopez, Robert H. Deng, and Feng Bao, editors, *ISC*, volume 3650 of *Lecture Notes in Computer Science*, pages 314–328. Springer, 2005.
- [LLL82] A. K. Lenstra, H. W. Lenstra, and L. Lovsz. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982. 10.1007/BF01457454.
- [LP11] Richard Lindner and Chris Peikert. Better key sizes (and attacks) for LWE-based encryption. In Aggelos Kiayias, editor, *CT-RSA*, volume 6558 of *Lecture Notes in Computer Science*, pages 319–339. Springer, 2011.
- [LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, pages 1–23, 2010. Draft of full version was provided by the authors.
- [MGH10] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with d -operand multiplications. In *CRYPTO*, pages 138–154, 2010.
- [Mic00] Daniele Micciancio. The shortest vector in a lattice is hard to approximate to within some constant. *SIAM J. Comput.*, 30(6):2008–2035, 2000.
- [Mic10] Daniele Micciancio. A first glimpse of cryptography’s holy grail. *Commun. ACM*, 53:96–96, March 2010.
- [MR09] Daniele Micciancio and Oded Regev. Lattice-based cryptography. In *Post-Quantum Cryptography*. Springer, 2009.
- [MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In Leonard J. Schulman, editor, *STOC*, pages 351–358. ACM, 2010.
- [OS07] Rafail Ostrovsky and William E. Skeith III. A survey of single-database private information retrieval: Techniques and applications. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 393–411. Springer, 2007.
- [Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342, 2009.
- [PVW08] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 554–571. Springer, 2008.

- [RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.
- [RS10] Markus Rückert and Michael Schneider. Estimating the security of lattice-based cryptosystems. Cryptology ePrint Archive, Report 2010/137, 2010. <http://eprint.iacr.org/>.
- [SS10] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394. Springer, 2010.
- [SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.
- [SY99] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for NC¹. In *FOCS*, pages 554–567, 1999.

Fully Homomorphic Encryption without Bootstrapping

Zvika Brakerski
Weizmann Institute of Science

Craig Gentry*
IBM T.J. Watson Research Center

Vinod Vaikuntanathan†
University of Toronto

Abstract

We present a radically new approach to fully homomorphic encryption (FHE) that dramatically improves performance and bases security on weaker assumptions. A central conceptual contribution in our work is a new way of constructing leveled fully homomorphic encryption schemes (capable of evaluating arbitrary polynomial-size circuits), *without Gentry's bootstrapping procedure*.

Specifically, we offer a choice of FHE schemes based on the learning with error (LWE) or ring-LWE (RLWE) problems that have 2^λ security against known attacks. For RLWE, we have:

- A leveled FHE scheme that can evaluate L -level arithmetic circuits with $\tilde{O}(\lambda \cdot L^3)$ per-gate computation – i.e., computation *quasi-linear* in the security parameter. Security is based on RLWE for an approximation factor exponential in L . This construction does not use the bootstrapping procedure.
- A leveled FHE scheme that uses bootstrapping *as an optimization*, where the per-gate computation (which includes the bootstrapping procedure) is $\tilde{O}(\lambda^2)$, *independent of L* . Security is based on the hardness of RLWE for *quasi-polynomial* factors (as opposed to the sub-exponential factors needed in previous schemes).

We obtain similar results for LWE, but with worse performance. We introduce a number of further optimizations to our schemes. As an example, for circuits of large width – e.g., where a constant fraction of levels have width at least λ – we can reduce the per-gate computation of the bootstrapped version to $\tilde{O}(\lambda)$, independent of L , by *batching the bootstrapping operation*. Previous FHE schemes all required $\tilde{\Omega}(\lambda^{3.5})$ computation per gate.

At the core of our construction is a much more effective approach for managing the noise level of lattice-based ciphertexts as homomorphic operations are performed, using some new techniques recently introduced by Brakerski and Vaikuntanathan (FOCS 2011).

*Sponsored by the Air Force Research Laboratory (AFRL). Disclaimer: This material is based on research sponsored by DARPA under agreement number FA8750-11-C-0096 and FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government. Approved for Public Release, Distribution Unlimited.

†This material is based on research sponsored by DARPA under Agreement number FA8750-11-2-0225. All disclaimers as above apply.

1 Introduction

Ancient History. Fully homomorphic encryption (FHE) [19, 8] allows a worker to receive encrypted data and perform arbitrarily-complex dynamically-chosen computations on that data while it remains encrypted, despite not having the secret decryption key. Until recently, all FHE schemes [8, 6, 20, 10, 5, 4] followed the same blueprint, namely the one laid out in Gentry’s original construction [8, 7].

The first step in Gentry’s blueprint is to construct a *somewhat homomorphic encryption (SWHE) scheme*, namely an encryption scheme capable of evaluating “low-degree” polynomials homomorphically. Starting with Gentry’s original construction based on ideal lattices [8], there are by now a number of such schemes in the literature [6, 20, 10, 5, 4, 13], all of which are based on lattices (either directly or implicitly). The ciphertexts in all these schemes are “noisy”, with a noise that grows slightly during homomorphic addition, and explosively during homomorphic multiplication, and hence, the limitation of low-degree polynomials.

To obtain FHE, Gentry provided a remarkable *bootstrapping theorem* which states that given a SWHE scheme that can evaluate its own decryption function (plus an additional operation), one can transform it into a “leveled”¹ FHE scheme. Bootstrapping “refreshes” a ciphertext by running the decryption function on it homomorphically, using an encrypted secret key (given in the public key), resulting in a reduced noise.

As if by a strange law of nature, SWHE schemes tend to be incapable of evaluating their own decryption circuits (plus some) without significant modifications. (We discuss recent exceptions [9, 3] below.) Thus, the final step is to *squash the decryption circuit* of the SWHE scheme, namely transform the scheme into one with the same homomorphic capacity but a decryption circuit that is simple enough to allow bootstrapping. Gentry [8] showed how to do this by adding a “hint” – namely, a large set with a secret sparse subset that sums to the original secret key – to the public key and relying on a “sparse subset sum” assumption.

1.1 Efficiency of Fully Homomorphic Encryption

The efficiency of fully homomorphic encryption has been a (perhaps, *the*) big question following its invention. In this paper, we are concerned with the *per-gate computation overhead* of the FHE scheme, defined as the ratio between the time it takes to compute a circuit homomorphically to the time it takes to compute it in the clear.² Unfortunately, FHE schemes that follow Gentry’s blueprint (some of which have actually been implemented [10, 5]) have fairly poor performance – their per-gate computation overhead is $p(\lambda)$, a large polynomial in the security parameter. In fact, we would like to argue that this penalty in performance is somewhat inherent for schemes that follow this blueprint.

First, the complexity of (known approaches to) bootstrapping is *inherently* at least the complexity of decryption *times* the bit-length of the individual ciphertexts that are used to encrypt the bits of the secret key. The reason is that bootstrapping involves evaluating the decryption circuit *homomorphically* – that is, in the decryption circuit, each secret-key bit is replaced by a (large) ciphertext that encrypts that bit – and both the complexity of decryption and the ciphertext lengths must each be $\Omega(\lambda)$.

Second, the undesirable properties of known SWHE schemes conspire to ensure that *the real cost of bootstrapping for FHE schemes that follow this blueprint is actually much worse than quadratic*. Known FHE schemes start with a SWHE scheme that can evaluate polynomials of degree D (multiplicative depth $\log D$) securely only if the underlying lattice problem is hard to 2^D -approximate in 2^λ time. For this to be hard, the lattice must have dimension $\Omega(D \cdot \lambda)$.³ Moreover, the coefficients of the vectors used in the

¹In a “leveled” FHE scheme, the size of the public key is linear in the *depth* of the circuits that the scheme can evaluate. One can obtain a “pure” FHE scheme (with a constant-size public key) from a leveled FHE scheme by assuming “circular security” – namely, that it is safe to encrypt the leveled FHE secret key under its own public key. We will omit the term “leveled” in this work.

²Other measures of efficiency, such ciphertext/key size and encryption/decryption time, are also important. In fact, the schemes we present in this paper are very efficient in these aspects (as are the schemes in [9, 3]).

³This is because we have lattice algorithms in n dimensions that compute $2^{n/\lambda}$ -approximations of short vectors in time $2^{\tilde{O}(\lambda)}$.

scheme have bit length $\Omega(D)$ to allow the ciphertext noise room to expand to 2^D . Therefore, the size of “fresh” ciphertexts (e.g., those that encrypt the bits of the secret key) is $\tilde{\Omega}(D^2 \cdot \lambda)$. Since the SWHE scheme must be “bootstrappable” – i.e., capable of evaluating its own decryption function – D must exceed the degree of the decryption function. Typically, the degree of the decryption function is $\Omega(\lambda)$. Thus, overall, “fresh” ciphertexts have size $\tilde{\Omega}(\lambda^3)$. So, the real cost of bootstrapping – even if we optimistically assume that the “stale” ciphertext that needs to be refreshed can be decrypted in only $\Theta(\lambda)$ -time – is $\tilde{\Omega}(\lambda^4)$.

The analysis above ignores a nice optimization by Stehlé and Steinfeld [22], which so far has not been useful in practice, that uses Chernoff bounds to asymptotically reduce the decryption degree down to $O(\sqrt{\lambda})$. With this optimization, the per-gate computation of FHE schemes that follow the blueprint is $\tilde{\Omega}(\lambda^3)$.⁴

Recent Deviations from Gentry’s Blueprint, and the Hope for Better Efficiency. Recently, Gentry and Halevi [9], and Brakerski and Vaikuntanathan [3], independently found very different ways to construct FHE without using the squashing step, and thus without the sparse subset sum assumption. These schemes are the first major deviations from Gentry’s blueprint for FHE. Brakerski and Vaikuntanathan [3] manage to base security entirely on LWE (for sub-exponential approximation factors), avoiding reliance on ideal lattices.

From an efficiency perspective, however, these results are not a clear win over previous schemes. Both of the schemes still rely on the problematic aspects of Gentry’s blueprint – namely, bootstrapping and an SWHE scheme with the undesirable properties discussed above. Thus, their per-gate computation is still $\tilde{\Omega}(\lambda^4)$ (in fact, that is an optimistic evaluation of their performance). Nevertheless, the techniques introduced in these recent constructions are very interesting and useful to us. In particular, we use the tools and techniques introduced by Brakerski and Vaikuntanathan [3] in an essential way to achieve remarkable efficiency gains.

An important, somewhat orthogonal question is the strength of assumptions underlying FHE schemes. All the schemes so far rely on the hardness of short vector problems on lattices with a *subexponential* approximation factor. Can we base FHE on polynomial hardness assumptions?

1.2 Our Results and Techniques

We leverage Brakerski and Vaikuntanathan’s techniques [3] to achieve asymptotically very efficient FHE schemes. Also, we base security on lattice problems with *quasi-polynomial* approximation factors. (Previous schemes all used sub-exponential factors.) In particular, we have the following theorem (informal):

- Assuming Ring LWE for an approximation factor exponential in L , we have a leveled FHE scheme that can evaluate L -level arithmetic circuits *without using bootstrapping*. The scheme has $\tilde{O}(\lambda \cdot L^3)$ per-gate computation (namely, *quasi-linear* in the security parameter).
- Alternatively, assuming Ring LWE is hard for *quasi-polynomial* factors, we have a leveled FHE scheme that uses bootstrapping *as an optimization*, where the per-gate computation (which includes the bootstrapping procedure) is $\tilde{O}(\lambda^2)$, *independent of L* .

We can alternatively base security on LWE, albeit with worse performance. We now sketch our main idea for boosting efficiency.

In the BV scheme [3], like ours, a ciphertext vector $\mathbf{c} \in R^n$ (where R is a ring, and n is the “dimension” of the vector) that encrypts a message m satisfies the decryption formula $m = \llbracket [\langle \mathbf{c}, \mathbf{s} \rangle]_q \rrbracket_2$, where $\mathbf{s} \in R^n$ is the secret key vector, q is an odd modulus, and $\llbracket \cdot \rrbracket_q$ denotes reduction into the range $(-q/2, q/2)$. This is an abstract scheme that can be instantiated with either LWE or Ring LWE – in the LWE instantiation, R is the ring of integers mod q and n is a large dimension, whereas in the Ring LWE instantiation, R is the ring of polynomials over integers mod q and an irreducible $f(x)$, and the dimension $n = 1$.

⁴We note that bootstrapping lazily – i.e., applying the refresh procedure only at a $1/k$ fraction of the circuit levels for $k > 1$ – cannot reduce the per-gate computation further by more than a logarithmic factor for schemes that follow this blueprint, since these SWHE schemes can evaluate only log multiplicative depth before it becomes *absolutely necessary* to refresh – i.e., $k = O(\log \lambda)$.

We will call $[\langle \mathbf{c}, \mathbf{s} \rangle]_q$ the *noise* associated to ciphertext \mathbf{c} under key \mathbf{s} . Decryption succeeds as long as the magnitude of the noise stays smaller than $q/2$. Homomorphic addition and multiplication increase the noise in the ciphertext. Addition of two ciphertexts with noise at most B results in a ciphertext with noise at most $2B$, whereas multiplication results in a noise as large as B^2 .⁵ We will describe a *noise-management technique* that keeps the noise in check by reducing it after homomorphic operations, without bootstrapping.

The key technical tool we use for noise management is the “modulus switching” technique developed by Brakerski and Vaikuntanathan [3]. Jumping ahead, we note that while they use modulus switching in “one shot” to obtain a small ciphertext (to which they then apply Gentry’s bootstrapping procedure), we will use it (iteratively, gradually) to keep the noise level essentially constant, while stingily sacrificing modulus size and gradually sacrificing the remaining homomorphic capacity of the scheme.

Modulus Switching. The essence of the modulus-switching technique is captured in the following lemma. In words, the lemma says that an evaluator, who does not know the secret key \mathbf{s} but instead only knows a bound on its length, can transform a ciphertext \mathbf{c} modulo q into a different ciphertext modulo p while preserving correctness – namely, $[\langle \mathbf{c}', \mathbf{s} \rangle]_p = [\langle \mathbf{c}, \mathbf{s} \rangle]_q \bmod 2$. The transformation from \mathbf{c} to \mathbf{c}' involves simply scaling by (p/q) and rounding appropriately! Most interestingly, if \mathbf{s} is short and p is sufficiently smaller than q , the “noise” in the ciphertext actually decreases – namely, $|\langle \mathbf{c}', \mathbf{s} \rangle|_p < |\langle \mathbf{c}, \mathbf{s} \rangle|_q$.

Lemma 1. *Let p and q be two odd moduli, and let \mathbf{c} be an integer vector. Define \mathbf{c}' to be the integer vector closest to $(p/q) \cdot \mathbf{c}$ such that $\mathbf{c}' = \mathbf{c} \bmod 2$. Then, for any \mathbf{s} with $|\langle \mathbf{c}, \mathbf{s} \rangle|_q < q/2 - (q/p) \cdot \ell_1(\mathbf{s})$, we have*

$$[\langle \mathbf{c}', \mathbf{s} \rangle]_p = [\langle \mathbf{c}, \mathbf{s} \rangle]_q \bmod 2 \quad \text{and} \quad |\langle \mathbf{c}', \mathbf{s} \rangle|_p < (p/q) \cdot |\langle \mathbf{c}, \mathbf{s} \rangle|_q + \ell_1(\mathbf{s})$$

where $\ell_1(\mathbf{s})$ is the ℓ_1 -norm of \mathbf{s} .

Proof. For some integer k , we have $[\langle \mathbf{c}, \mathbf{s} \rangle]_q = \langle \mathbf{c}, \mathbf{s} \rangle - kq$. For the same k , let $e_p = \langle \mathbf{c}', \mathbf{s} \rangle - kp \in \mathbb{Z}$. Since $\mathbf{c}' = \mathbf{c}$ and $p = q \bmod 2$, we have $e_p = [\langle \mathbf{c}, \mathbf{s} \rangle]_q \bmod 2$. Therefore, to prove the lemma, it suffices to prove that $e_p = [\langle \mathbf{c}', \mathbf{s} \rangle]_p$ and that it has small enough norm. We have $e_p = (p/q)[\langle \mathbf{c}, \mathbf{s} \rangle]_q + \langle \mathbf{c}' - (p/q)\mathbf{c}, \mathbf{s} \rangle$, and therefore $|e_p| \leq (p/q)|[\langle \mathbf{c}, \mathbf{s} \rangle]_q| + \ell_1(\mathbf{s}) < p/2$. The latter inequality implies $e_p = [\langle \mathbf{c}', \mathbf{s} \rangle]_p$. \square

Amazingly, this trick permits the evaluator to reduce the magnitude of the noise without knowing the secret key, and without bootstrapping. In other words, modulus switching gives us a very powerful and lightweight way to *manage the noise* in FHE schemes! In [3], the modulus switching technique is bundled into a “dimension reduction” procedure, and we believe it deserves a separate name and close scrutiny. It is also worth noting that our use of modulus switching does not require an “evaluation key”, in contrast to [3].

Our New Noise Management Technique. At first, it may look like modulus switching is not a very effective noise management tool. If p is smaller than q , then of course modulus switching may reduce the magnitude of the noise, but it reduces the modulus size by essentially the same amount. In short, the ratio of the noise to the “noise ceiling” (the modulus size) does not decrease at all. Isn’t this ratio what dictates the remaining homomorphic capacity of the scheme, and how can potentially worsening (certainly not improving) this ratio do anything useful?

In fact, it’s not just the ratio of the noise to the “noise ceiling” that’s important. The *absolute magnitude of the noise* is also important, especially in multiplications. Suppose that $q \approx x^k$, and that you have two mod- q SWHE ciphertexts with noise of magnitude x . If you multiply them, the noise becomes x^2 . After 4 levels of multiplication, the noise is x^{16} . If you do another multiplication at this point, you reduce the ratio of the noise ceiling (i.e. q) to the noise level by a huge factor of x^{16} – i.e., you reduce this gap very

⁵The noise after multiplication is in fact a bit larger than B^2 due to the additional noise from the BV “re-linearization” process. For the purposes of this exposition, it is best to ignore this minor detail.

fast. Thus, the actual magnitude of the noise impacts how fast this gap is reduced. After only $\log k$ levels of multiplication, the noise level reaches the ceiling.

Now, consider the following alternative approach. Choose a *ladder of gradually decreasing moduli* $\{q_i \approx q/x^i\}$ for $i < k$. After you multiply the two mod- q ciphertexts, switch the ciphertext to the smaller modulus $q_1 = q/x$. As the lemma above shows, the noise level of the new ciphertext (now with respect to the modulus q_1) goes from x^2 back down to x . (Let's suppose for now that $\ell_1(s)$ is small in comparison to x so that we can ignore it.) Now, when we multiply two ciphertexts (wrt modulus q_1) that have noise level x , the noise again becomes x^2 , but then we switch to modulus q_2 to reduce the noise back to x . In short, each level of multiplication only reduces the ratio (noise ceiling)/(noise level) by a factor of x (not something like x^{16}). With this new approach, we can perform about k (not just $\log k$) levels of multiplication before we reach the noise ceiling. We have just increased (without bootstrapping) the number of multiplicative levels that we can evaluate by an exponential factor!

This exponential improvement is enough to achieve leveled FHE without bootstrapping. For *any* polynomial k , we can evaluate circuits of depth k . The performance of the scheme degrades with k – e.g., we need to set $q = q_0$ to have bit length proportional to k – but it degrades only polynomially with k .

Our main observation – the key to obtaining FHE without bootstrapping – is so simple that it is easy to miss and bears repeating: We get noise reduction *automatically* via modulus switching, and by carefully calibrating our ladder of moduli $\{q_i\}$, one modulus for each circuit level, to be decreasing *gradually*, we can keep the noise level very small and essentially constant from one level to the next while only gradually sacrificing the size of our modulus until the ladder is used up. With this approach, we can efficiently evaluate arbitrary polynomial-size arithmetic circuits without resorting to bootstrapping.

Performance-wise, this scheme trounces previous (bootstrapping-based) FHE schemes (at least asymptotically; the concrete performance remains to be seen). Instantiated with ring-LWE, it can evaluate L -level arithmetic circuits with per-gate computation $\tilde{O}(\lambda \cdot L^3)$ – i.e., computation *quasi-linear* in the security parameter. Since the ratio of the largest modulus (namely, $q \approx x^L$) to the noise (namely, x) is exponential in L , the scheme relies on the hardness of approximating short vectors to within an exponential in L factor.

Bootstrapping for Better Efficiency and Better Assumptions. The per-gate computation of our FHE-without-bootstrapping scheme depends polynomially on the number of levels in the circuit that is being evaluated. While this approach is efficient (in the sense of “polynomial time”) for polynomial-size circuits, the per-gate computation may become undesirably high for very deep circuits. So, we re-introduce bootstrapping *as an optimization*⁶ that makes the per-gate computation independent of the circuit depth, and that (if one is willing to assume circular security) allows homomorphic operations to be performed indefinitely without needing to specify in advance a bound on the number of circuit levels. The main idea is that to compute arbitrary polynomial-depth circuits, it is enough to compute the decryption circuit of the scheme homomorphically. Since the decryption circuit has depth $\approx \log \lambda$, the largest modulus we need has only $\tilde{O}(\lambda)$ bits, and therefore we can base security on the hardness of lattice problems with quasi-polynomial factors. Since the decryption circuit has size $\tilde{O}(\lambda)$ for the RLWE-based instantiation, the per-gate computation becomes $\tilde{O}(\lambda^2)$ (independent of L). See Section 5 for details.

Other Optimizations. We also consider *batching* as an optimization. The idea behind batching is to pack multiple plaintexts into each ciphertext so that a function can be homomorphically evaluated on multiple inputs with approximately the same efficiency as homomorphically evaluating it on one input.

⁶We are aware of the seeming irony of trumpeting “FHE without bootstrapping” and then proposing bootstrapping “as an optimization”. First, FHE without bootstrapping is exciting theoretically, independent of performance. Second, whether bootstrapping actually improves performance depends crucially on the number of levels in the circuit one is evaluating. For example, for circuits of depth sub-polynomial in the security parameter, this “optimization” will not improve performance asymptotically.

An especially interesting case is *batching the decryption function* so that multiple ciphertexts – e.g., all of the ciphertexts associated to gates at some level in the circuit – can be bootstrapped simultaneously very efficiently. For circuits of large width (say, width λ), batched bootstrapping reduces the per-gate computation in the RLWE-based instantiation to $\tilde{O}(\lambda)$, independent of L . We give the details in Section 5.

1.3 Other Related Work

We note that prior to Gentry’s construction, there were already a few interesting homomorphic encryptions schemes that could be called “somewhat homomorphic”, including Boneh-Goh-Nissim [2] (evaluates quadratic formulas using bilinear maps), (Aguilar Melchor)-Gaborit-Herranz [15] (evaluates constant degree polynomials using lattices) and Ishai-Paskin [12] (evaluates branching programs).

2 Preliminaries

Basic Notation. In our construction, we will use a ring R . In our concrete instantiations, we prefer to use either $R = \mathbb{Z}$ (the integers) or the polynomial ring $R = \mathbb{Z}[x]/(x^d + 1)$, where d is a power of 2.

We write elements of R in lowercase – e.g., $r \in R$. We write vectors in bold – e.g., $\mathbf{v} \in R^n$. The notation $\mathbf{v}[i]$ refers to the i -th coefficient of \mathbf{v} . We write the dot product of $\mathbf{u}, \mathbf{v} \in R^n$ as $\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{i=1}^n \mathbf{u}[i] \cdot \mathbf{v}[i] \in R$. When R is a polynomial ring, $\|r\|$ for $r \in R$ refers to the Euclidean norm of r ’s coefficient vector. We say $\gamma_R = \max\{\|a \cdot b\|/\|a\|\|b\| : a, b \in R\}$ is the expansion factor of R . For $R = \mathbb{Z}[x]/(x^d + 1)$, the value of γ_R is at most \sqrt{d} by Cauchy-Schwarz.

For integer q , we use R_q to denote R/qR . Sometimes we will use abuse notation and use R_2 to denote the set of R -elements with binary coefficients – e.g., when $R = \mathbb{Z}$, R_2 may denote $\{0, 1\}$, and when R is a polynomial ring, R_2 may denote those polynomials that have 0/1 coefficients. When it is obvious that q is not a power of two, we will use $\lceil \log q \rceil$ to denote $1 + \lfloor \log q \rfloor$. For $a \in R$, we use the notation $[a]_q$ to refer to $a \bmod q$, with coefficients reduced into the range $(-q/2, q/2]$.

Leveled Fully Homomorphic Encryption. Most of this paper will focus on the construction of a *leveled* fully homomorphic scheme, in the sense that the parameters of the scheme depend (polynomially) on the depth of the circuits that the scheme is capable of evaluating.

Definition 1 (Leveled Fully Homomorphic Encryption [7]). *We say that a family of homomorphic encryption schemes $\{\mathcal{E}^{(L)} : L \in \mathbb{Z}^+\}$ is leveled fully homomorphic if, for all $L \in \mathbb{Z}^+$, they all use the same decryption circuit, $\mathcal{E}^{(L)}$ compactly evaluates all circuits of depth at most L (that use some specified complete set of gates), and the computational complexity of $\mathcal{E}^{(L)}$ ’s algorithms is polynomial (the same polynomial for all L) in the security parameter, L , and (in the case of the evaluation algorithm) the size of the circuit.*

2.1 The Learning with Errors (LWE) Problem

The learning with errors (LWE) problem was introduced by Regev [17]. It is defined as follows.

Definition 2 (LWE). *For security parameter λ , let $n = n(\lambda)$ be an integer dimension, let $q = q(\lambda) \geq 2$ be an integer, and let $\chi = \chi(\lambda)$ be a distribution over \mathbb{Z} . The $\text{LWE}_{n,q,\chi}$ problem is to distinguish the following two distributions: In the first distribution, one samples (\mathbf{a}_i, b_i) uniformly from \mathbb{Z}_q^{n+1} . In the second distribution, one first draws $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ uniformly and then samples $(\mathbf{a}_i, b_i) \in \mathbb{Z}_q^{n+1}$ by sampling $\mathbf{a}_i \leftarrow \mathbb{Z}_q^n$ uniformly, $e_i \leftarrow \chi$, and setting $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$. The $\text{LWE}_{n,q,\chi}$ assumption is that the $\text{LWE}_{n,q,\chi}$ problem is infeasible.*

Regev [17] proved that for certain moduli q and Gaussian error distributions χ , the $\text{LWE}_{n,q,\chi}$ assumption is true as long as certain worst-case lattice problems are hard to solve using a quantum algorithm. We state this result using the terminology of B -bounded distributions, which is a distribution over the integers where the magnitude of a sample is bounded with high probability. A definition follows.

Definition 3 (*B*-bounded distributions). A distribution ensemble $\{\chi_n\}_{n \in \mathbb{N}}$, supported over the integers, is called *B*-bounded if

$$\Pr_{e \leftarrow \chi_n} [|e| > B] = \text{negl}(n) .$$

We can now state Regev’s worst-case to average-case reduction for LWE.

Theorem 1 (Regev [17]). For any integer dimension n , prime integer $q = q(n)$, and $B = B(n) \geq 2n$, there is an efficiently samplable *B*-bounded distribution χ such that if there exists an efficient (possibly quantum) algorithm that solves $\text{LWE}_{n,q,\chi}$, then there is an efficient quantum algorithm for solving $\tilde{O}(qn^{1.5}/B)$ -approximate worst-case SVP and gapSVP.

Peikert [16] de-quantized Regev’s results to some extent – that is, he showed the $\text{LWE}_{n,q,\chi}$ assumption is true as long as certain worst-case lattice problems are hard to solve using a *classical* algorithm. (See [16] for a precise statement of these results.)

Applebaum et al. [1] showed that if LWE is hard for the above distribution of s , then it is also hard when s ’s coefficients are sampled according to the noise distribution χ .

2.2 The Ring Learning with Errors (RLWE) Problem

The ring learning with errors (RLWE) problem was introduced by Lyubashevsky, Peikert and Regev [14]. We will use an simplified special-case version of the problem that is easier to work with [18, 4].

Definition 4 (RLWE). For security parameter λ , let $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2. Let $q = q(\lambda) \geq 2$ be an integer. Let $R = \mathbb{Z}[x]/(f(x))$ and let $R_q = R/qR$. Let $\chi = \chi(\lambda)$ be a distribution over R . The $\text{RLWE}_{d,q,\chi}$ problem is to distinguish the following two distributions: In the first distribution, one samples (a_i, b_i) uniformly from R_q^2 . In the second distribution, one first draws $s \leftarrow R_q$ uniformly and then samples $(a_i, b_i) \in R_q^2$ by sampling $a_i \leftarrow R_q$ uniformly, $e_i \leftarrow \chi$, and setting $b_i = a_i \cdot s + e_i$. The $\text{RLWE}_{d,q,\chi}$ assumption is that the $\text{RLWE}_{d,q,\chi}$ problem is infeasible.

The RLWE problem is useful, because the well-established shortest vector problem (SVP) over ideal lattices can be reduced to it, specifically:

Theorem 2 (Lyubashevsky-Peikert-Regev [14]). For any d that is a power of 2, ring $R = \mathbb{Z}[x]/(x^d + 1)$, prime integer $q = q(d) = 1 \bmod d$, and $B = \omega(\sqrt{d} \log d)$, there is an efficiently samplable distribution χ that outputs elements of R of length at most B with overwhelming probability, such that if there exists an efficient algorithm that solves $\text{RLWE}_{d,q,\chi}$, then there is an efficient quantum algorithm for solving $d^{\omega(1)} \cdot (q/B)$ -approximate worst-case SVP for ideal lattices over R .

Typically, to use RLWE with a cryptosystem, one chooses the noise distribution χ according to a Gaussian distribution, where vectors sampled according to this distribution have length only $\text{poly}(d)$ with overwhelming probability. This Gaussian distribution may need to be “ellipsoidal” for certain reductions to go through [14]. It has been shown for RLWE that one can equivalently assume that s is alternatively sampled from the noise distribution χ [14].

2.3 The General Learning with Errors (GLWE) Problem

The learning with errors (LWE) problem and the ring learning with errors problem RLWE are syntactically identical, aside from using different rings (\mathbb{Z} versus a polynomial ring) and different vector dimensions over those rings ($n = \text{poly}(\lambda)$ for LWE, but n is constant – namely, 1 – in the RLWE case). To simplify our presentation, we define a “General Learning with Errors (GLWE)” Problem, and describe a single “GLWE-based” FHE scheme, rather than presenting essentially the same scheme twice, once for each of our two concrete instantiations.

Definition 5 (GLWE). For security parameter λ , let $n = n(\lambda)$ be an integer dimension, let $f(x) = x^d + 1$ where $d = d(\lambda)$ is a power of 2, let $q = q(\lambda) \geq 2$ be a prime integer, let $R = \mathbb{Z}[x]/(f(x))$ and $R_q = R/qR$, and let $\chi = \chi(\lambda)$ be a distribution over R . The $\text{GLWE}_{n,f,q,\chi}$ problem is to distinguish the following two distributions: In the first distribution, one samples (\mathbf{a}_i, b_i) uniformly from R_q^{n+1} . In the second distribution, one first draws $\mathbf{s} \leftarrow R_q^n$ uniformly and then samples $(\mathbf{a}_i, b_i) \in R_q^{n+1}$ by sampling $\mathbf{a}_i \leftarrow R_q^n$ uniformly, $e_i \leftarrow \chi$, and setting $b_i = \langle \mathbf{a}_i, \mathbf{s} \rangle + e_i$. The $\text{GLWE}_{n,f,q,\chi}$ assumption is that the $\text{GLWE}_{n,f,q,\chi}$ problem is infeasible.

LWE is simply GLWE instantiated with $d = 1$. RLWE is GLWE instantiated with $n = 1$. Interestingly, as far as we know, instances of GLWE between these extremes have not been explored. One would suspect that GLWE is hard for any (n, d) such that $n \cdot d = \Omega(\lambda \log(q/B))$, where B is a bound (with overwhelming probability) on the length of elements output by χ . For fixed $n \cdot d$, perhaps GLWE gradually becomes harder as n increases (if it is true that general lattice problems are harder than ideal lattice problems), whereas increasing d is probably often preferable for efficiency.

If q is much larger than B , the associated GLWE problem is believed to be easier (i.e., there is less security). Previous FHE schemes required q/B to be sub-exponential in n or d to give room for the noise to grow as homomorphic operations (especially multiplication) are performed. In our FHE scheme without bootstrapping, q/B will be exponential in the number of circuit levels to be evaluated. However, since the decryption circuit can be evaluated in logarithmic depth, the bootstrapped version of our scheme will only need q/B to be quasi-polynomial, and we thus base security on lattice problems for quasi-polynomial approximation factors.

The GLWE assumption implies that the distribution $\{(\mathbf{a}_i, \langle \mathbf{a}_i, \mathbf{s} \rangle + t \cdot e_i)\}$ is computational indistinguishable from uniform for any t relatively prime to q . This fact will be convenient for encryption, where, for example, a message m may be encrypted as $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + 2e + m)$, and this fact can be used to argue that the second component of this message is indistinguishable from random.

3 (Leveled) FHE without Bootstrapping: Our Construction

The plan of this section is to present our leveled FHE-without-bootstrapping construction in modular steps. First, we describe a plain GLWE-based encryption scheme with no homomorphic operations. Next, we describe variants of the “relinearization” and “dimension reduction” techniques of [3]. Finally, in Section 3.4, we lay out our construction of FHE without bootstrapping.

3.1 Basic Encryption Scheme

We begin by presenting a basic GLWE-based encryption scheme with no homomorphic operations. Let λ be the security parameter, representing 2^λ security against known attacks. ($\lambda = 100$ is a reasonable value.)

Let $R = R(\lambda)$ be a ring. For example, one may use $R = \mathbb{Z}$ if one wants a scheme based on (standard) LWE, or one may use $R = \mathbb{Z}[x]/f(x)$ where (e.g.) $f(x) = x^d + 1$ and $d = d(\lambda)$ is a power of 2 if one wants a scheme based on RLWE. Let the “dimension” $n = n(\lambda)$, an odd modulus $q = q(\lambda)$, a “noise” distribution $\chi = \chi(\lambda)$ over R , and an integer $N = N(\lambda)$ be additional parameters of the system. These parameters come from the GLWE assumption, except for N , which is set to be larger than $(2n + 1) \log q$. Note that $n = 1$ in the RLWE instantiation. For simplicity, assume for now that the plaintext space is $R_2 = R/2R$, though larger plaintext spaces are certainly possible.

We go ahead and stipulate here – even though it only becomes important when we introduce homomorphic operations – that the noise distribution χ is set to be as small as possible. Specifically, to base security on LWE or GLWE, one must use (typically Gaussian) noise distributions with deviation *at least* some sub-linear function of d or n , and we will let χ be a noise distribution that barely satisfies that requirement. To

achieve 2^λ security against known lattice attacks, one must have $n \cdot d = \Omega(\lambda \cdot \log(q/B))$ where B is a bound on the length of the noise. Since n or d depends logarithmically on q , and since the distribution χ (and hence B) depends sub-linearly on n or d , the distribution χ (and hence B) depends sub-logarithmically on q . This dependence is weak, and one should think of the noise distribution as being essentially independent of q .

Here is a basic GLWE-based encryption scheme with no homomorphic operations:

Basic GLWE-Based Encryption Scheme:

- **E.Setup**($1^\lambda, 1^\mu, b$): Use the bit $b \in \{0, 1\}$ to determine whether we are setting parameters for a LWE-based scheme (where $d = 1$) or a RLWE-based scheme (where $n = 1$). Choose a μ -bit modulus q and choose the other parameters ($d = d(\lambda, \mu, b)$, $n = n(\lambda, \mu, b)$, $N = \lceil (2n + 1) \log q \rceil$, $\chi = \chi(\lambda, \mu, b)$) appropriately to ensure that the scheme is based on a GLWE instance that achieves 2^λ security against known attacks. Let $R = \mathbb{Z}[x]/(x^d + 1)$ and let $params = (q, d, n, N, \chi)$.
- **E.SecretKeyGen**($params$): Draw $s' \leftarrow \chi^n$. Set $sk = s \leftarrow (1, s'[1], \dots, s'[n]) \in R_q^{n+1}$.
- **E.PublicKeyGen**($params, sk$): Takes as input a secret key $sk = s = (1, s')$ with $s[0] = 1$ and $s' \in R_q^n$ and the $params$. Generate matrix $A' \leftarrow R_q^{N \times n}$ uniformly and a vector $e \leftarrow \chi^N$ and set $b \leftarrow A's' + 2e$. Set A to be the $(n + 1)$ -column matrix consisting of b followed by the n columns of $-A'$. (Observe: $A \cdot s = 2e$.) Set the public key $pk = A$.
- **E.Enc**($params, pk, m$): To encrypt a message $m \in R_2$, set $\mathbf{m} \leftarrow (m, 0, \dots, 0) \in R_q^{n+1}$, sample $\mathbf{r} \leftarrow R_2^N$ and output the ciphertext $\mathbf{c} \leftarrow \mathbf{m} + A^T \mathbf{r} \in R_q^{n+1}$.
- **E.Dec**($params, sk, \mathbf{c}$): Output $m \leftarrow \llbracket \langle \mathbf{c}, s \rangle \rrbracket_q \rfloor_2$.

Correctness is easy to see, and it is straightforward to base security on special cases (depending on the parameters) of the GLWE assumption (and one can find such proofs of special cases in prior work).

3.2 Key Switching (Dimension Reduction)

We start by reminding the reader that in the basic GLWE-based encryption scheme above, the decryption equation for a ciphertext \mathbf{c} that encrypts m under key s can be written as $m = \llbracket L_{\mathbf{c}}(s) \rrbracket_q \rfloor_2$ where $L_{\mathbf{c}}(\mathbf{x})$ is a ciphertext-dependent linear equation over the coefficients of \mathbf{x} given by $L_{\mathbf{c}}(\mathbf{x}) = \langle \mathbf{c}, \mathbf{x} \rangle$.

Suppose now that we have two ciphertexts \mathbf{c}_1 and \mathbf{c}_2 , encrypting m_1 and m_2 respectively under the same secret key s . The way homomorphic multiplication is accomplished in [3] is to consider the *quadratic* equation $Q_{\mathbf{c}_1, \mathbf{c}_2}(\mathbf{x}) \leftarrow L_{\mathbf{c}_1}(\mathbf{x}) \cdot L_{\mathbf{c}_2}(\mathbf{x})$. Assuming the noises of the initial ciphertexts are small enough, we obtain $m_1 \cdot m_2 = \llbracket Q_{\mathbf{c}_1, \mathbf{c}_2}(s) \rrbracket_q \rfloor_2$, as desired. If one wishes, one can view $Q_{\mathbf{c}_1, \mathbf{c}_2}(\mathbf{x})$ as a *linear* equation $L_{\mathbf{c}_1, \mathbf{c}_2}^{long}(\mathbf{x} \otimes \mathbf{x})$ over the coefficients of $\mathbf{x} \otimes \mathbf{x}$ – that is, the tensoring of \mathbf{x} with itself – where $\mathbf{x} \otimes \mathbf{x}$'s dimension is roughly the square of \mathbf{x} 's. Using this interpretation, the ciphertext represented by the coefficients of the linear equation L^{long} is decryptable by the long secret key $s_1 \otimes s_1$ via the usual dot product. Of course, we cannot continue increasing the dimension like this indefinitely and preserve efficiency.

Thus, Brakerski and Vaikuntanathan convert the long ciphertext represented by the linear equation L^{long} and decryptable by the long tensored secret key $s_1 \otimes s_1$ into a shorter ciphertext \mathbf{c}_2 that is decryptable by a different secret key s_2 . (The secret keys need to be different to avoid a “circular security” issue). Encryptions of $s_1 \otimes s_1$ under s_2 are provided in the public key as a “hint” to facilitate this conversion.

We observe that Brakerski and Vaikuntanathan's relinearization / dimension reduction procedures are actually quite a bit more general. They can be used to not only reduce the dimension of the ciphertext, but more generally, can be used to transform a ciphertext \mathbf{c}_1 that is decryptable under one secret key vector s_1 to

a different ciphertext \mathbf{c}_2 that encrypts the same message, but is now decryptable under a second secret key vector \mathbf{s}_2 . The vectors $\mathbf{c}_2, \mathbf{s}_2$ may not necessarily be of lower degree or dimension than $\mathbf{c}_1, \mathbf{s}_1$.

Below, we review the concrete details of Brakerski and Vaikuntanathan's key switching procedures. The procedures will use some subroutines that, given two vectors \mathbf{c} and \mathbf{s} , "expand" these vectors to get longer (higher-dimensional) vectors \mathbf{c}' and \mathbf{s}' such that $\langle \mathbf{c}', \mathbf{s}' \rangle = \langle \mathbf{c}, \mathbf{s} \rangle \bmod q$. We describe these subroutines first.

- $\text{BitDecomp}(\mathbf{x} \in R_q^n, q)$ decomposes \mathbf{x} into its bit representation. Namely, write $\mathbf{x} = \sum_{j=0}^{\lfloor \log q \rfloor} 2^j \cdot \mathbf{u}_j$, where all of the vectors \mathbf{u}_j are in R_2^n , and output $(\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{\lfloor \log q \rfloor}) \in R_2^{n \cdot \lceil \log q \rceil}$.
- $\text{Powersof2}(\mathbf{x} \in R_q^n, q)$ outputs the vector $(\mathbf{x}, 2 \cdot \mathbf{x}, \dots, 2^{\lfloor \log q \rfloor} \cdot \mathbf{x}) \in R_q^{n \cdot \lceil \log q \rceil}$.

If one knows *a priori* that \mathbf{x} has coefficients in $[0, B]$ for $B \ll q$, then BitDecomp can be optimized in the obvious way to output a shorter decomposition in $R_2^{n \cdot \lceil \log B \rceil}$. Observe that:

Lemma 2. *For vectors \mathbf{c}, \mathbf{s} of equal length, we have $\langle \text{BitDecomp}(\mathbf{c}, q), \text{Powersof2}(\mathbf{s}, q) \rangle = \langle \mathbf{c}, \mathbf{s} \rangle \bmod q$.*

Proof.

$$\langle \text{BitDecomp}(\mathbf{c}, q), \text{Powersof2}(\mathbf{s}, q) \rangle = \sum_{j=0}^{\lfloor \log q \rfloor} \langle \mathbf{u}_j, 2^j \cdot \mathbf{s} \rangle = \sum_{j=0}^{\lfloor \log q \rfloor} \langle 2^j \cdot \mathbf{u}_j, \mathbf{s} \rangle = \left\langle \sum_{j=0}^{\lfloor \log q \rfloor} 2^j \cdot \mathbf{u}_j, \mathbf{s} \right\rangle = \langle \mathbf{c}, \mathbf{s} \rangle .$$

□

We remark that this obviously generalizes to decompositions wrt bases other than the powers of 2.

Now, key switching consists of two procedures: first, a procedure $\text{SwitchKeyGen}(\mathbf{s}_1, \mathbf{s}_2, n_1, n_2, q)$, which takes as input the two secret key vectors as input, the respective dimensions of these vectors, and the modulus q , and outputs some auxiliary information $\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}$ that enables the switching; and second, a procedure $\text{SwitchKey}(\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}, \mathbf{c}_1, n_1, n_2, q)$, that takes this auxiliary information and a ciphertext encrypted under \mathbf{s}_1 and outputs a new ciphertext \mathbf{c}_2 that encrypts the same message under the secret key \mathbf{s}_2 . (Below, we often suppress the additional arguments n_1, n_2, q .)

$\text{SwitchKeyGen}(\mathbf{s}_1 \in R_q^{n_1}, \mathbf{s}_2 \in R_q^{n_2})$:

1. Run $\mathbf{A} \leftarrow \text{E.PublicKeyGen}(\mathbf{s}_2, N)$ for $N = n_1 \cdot \lceil \log q \rceil$.
2. Set $\mathbf{B} \leftarrow \mathbf{A} + \text{Powersof2}(\mathbf{s}_1)$ (Add $\text{Powersof2}(\mathbf{s}_1) \in R_q^N$ to \mathbf{A} 's first column.) Output $\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2} = \mathbf{B}$.

$\text{SwitchKey}(\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}, \mathbf{c}_1)$: Output $\mathbf{c}_2 = \text{BitDecomp}(\mathbf{c}_1)^T \cdot \mathbf{B} \in R_q^{n_2}$.

Note that, in SwitchKeyGen , the matrix \mathbf{A} basically consists of encryptions of 0 under the key \mathbf{s}_2 . Then, pieces of the key \mathbf{s}_1 are added to these encryptions of 0. Thus, in some sense, the matrix \mathbf{B} consists of encryptions of pieces of \mathbf{s}_1 (in a certain format) under the key \mathbf{s}_2 . We now establish that the key switching procedures are meaningful, in the sense that they preserve the correctness of decryption under the new key.

Lemma 3. [Correctness] *Let $\mathbf{s}_1, \mathbf{s}_2, q, n_1, n_2, \mathbf{A}, \mathbf{B} = \tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}$ be as in $\text{SwitchKeyGen}(\mathbf{s}_1, \mathbf{s}_2)$, and let $\mathbf{A} \cdot \mathbf{s}_2 = 2\mathbf{e}_2 \in R_q^N$. Let $\mathbf{c}_1 \in R_q^{n_1}$ and $\mathbf{c}_2 \leftarrow \text{SwitchKey}(\tau_{\mathbf{s}_1 \rightarrow \mathbf{s}_2}, \mathbf{c}_1)$. Then,*

$$\langle \mathbf{c}_2, \mathbf{s}_2 \rangle = 2 \langle \text{BitDecomp}(\mathbf{c}_1), \mathbf{e}_2 \rangle + \langle \mathbf{c}_1, \mathbf{s}_1 \rangle \bmod q$$

Proof.

$$\begin{aligned}
\langle \mathbf{c}_2, \mathbf{s}_2 \rangle &= \text{BitDecomp}(\mathbf{c}_1)^T \cdot \mathbf{B} \cdot \mathbf{s}_2 \\
&= \text{BitDecomp}(\mathbf{c}_1)^T \cdot (2\mathbf{e}_2 + \text{Powersof2}(\mathbf{s}_1)) \\
&= 2 \langle \text{BitDecomp}(\mathbf{c}_1), \mathbf{e}_2 \rangle + \langle \text{BitDecomp}(\mathbf{c}_1), \text{Powersof2}(\mathbf{s}_1) \rangle \\
&= 2 \langle \text{BitDecomp}(\mathbf{c}_1), \mathbf{e}_2 \rangle + \langle \mathbf{c}_1, \mathbf{s}_1 \rangle
\end{aligned}$$

□

Note that the dot product of $\text{BitDecomp}(\mathbf{c}_1)$ and \mathbf{e}_2 is small, since $\text{BitDecomp}(\mathbf{c}_1)$ is in R_2^N . Overall, we have that \mathbf{c}_2 is a valid encryption of m under key \mathbf{s}_2 , with noise that is larger by a small additive factor.

3.3 Modulus Switching

Suppose \mathbf{c} is a valid encryption of m under \mathbf{s} modulo q (i.e., $m = \llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q$), and that \mathbf{s} is a *short* vector. Suppose also that \mathbf{c}' is basically a simple scaling of \mathbf{c} – in particular, \mathbf{c}' is the R -vector closest to $(p/q) \cdot \mathbf{c}$ such that $\mathbf{c}' = \mathbf{c} \bmod 2$. Then, it turns out (subject to some qualifications) that \mathbf{c}' is a valid encryption of m under \mathbf{s} modulo p using the usual decryption equation – that is, $m = \llbracket \langle \mathbf{c}', \mathbf{s} \rangle \rrbracket_p$! In other words, we can change the inner modulus in the decryption equation – e.g., to a smaller number – while preserving the correctness of decryption under the same secret key! The essence of this modulus switching idea, a variant of Brakerski and Vaikuntanathan’s modulus reduction technique, is formally captured in Lemma 4 below.

Definition 6 (Scale). For integer vector \mathbf{x} and integers $q > p > m$, we define $\mathbf{x}' \leftarrow \text{Scale}(\mathbf{x}, q, p, r)$ to be the R -vector closest to $(p/q) \cdot \mathbf{x}$ that satisfies $\mathbf{x}' = \mathbf{x} \bmod r$.

Definition 7 ($\ell_1^{(R)}$ norm). The (usual) norm $\ell_1(\mathbf{s})$ over the reals equals $\sum_i \|\mathbf{s}[i]\|$. We extend this to our ring R as follows: $\ell_1^{(R)}(\mathbf{s})$ for $\mathbf{s} \in R^n$ is defined as $\sum_i \|\mathbf{s}[i]\|$.

Lemma 4. Let d be the degree of the ring (e.g., $d = 1$ when $R = \mathbb{Z}$). Let $q > p > r$ be positive integers satisfying $q = p = 1 \bmod r$. Let $\mathbf{c} \in R^n$ and $\mathbf{c}' \leftarrow \text{Scale}(\mathbf{c}, q, p, r)$. Then, for any $\mathbf{s} \in R^n$ with $\|\llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q\| < q/2 - (q/p) \cdot (r/2) \cdot \sqrt{d} \cdot \gamma(R) \cdot \ell_1^{(R)}(\mathbf{s})$, we have

$$\llbracket \langle \mathbf{c}', \mathbf{s} \rangle \rrbracket_p = \llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q \bmod r \quad \text{and} \quad \|\langle \mathbf{c}', \mathbf{s} \rangle\| < (p/q) \cdot \|\llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q\| + (r/2) \cdot \sqrt{d} \cdot \gamma(R) \cdot \ell_1^{(R)}(\mathbf{s})$$

Proof. (Lemma 4) We have

$$\llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q = \langle \mathbf{c}, \mathbf{s} \rangle - kq$$

for some $k \in R$. For the same k , let

$$e_p = \langle \mathbf{c}', \mathbf{s} \rangle - kp \in R$$

Note that $e_p = \llbracket \langle \mathbf{c}', \mathbf{s} \rangle \rrbracket_p \bmod p$. We claim that $\|e_p\|$ is so small that $e_p = \llbracket \langle \mathbf{c}', \mathbf{s} \rangle \rrbracket_p$. We have:

$$\begin{aligned}
\|e_p\| &= \| -kp + \langle (p/q) \cdot \mathbf{c}, \mathbf{s} \rangle + \langle \mathbf{c}' - (p/q) \cdot \mathbf{c}, \mathbf{s} \rangle \| \\
&\leq \| -kp + \langle (p/q) \cdot \mathbf{c}, \mathbf{s} \rangle \| + \| \langle \mathbf{c}' - (p/q) \cdot \mathbf{c}, \mathbf{s} \rangle \| \\
&\leq (p/q) \cdot \|\llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q\| + \gamma(R) \cdot \sum_{j=1}^n \|\mathbf{c}'[j] - (p/q) \cdot \mathbf{c}[j]\| \cdot \|\mathbf{s}[j]\| \\
&\leq (p/q) \cdot \|\llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q\| + \gamma(R) \cdot (r/2) \cdot \sqrt{d} \cdot \ell_1^{(R)}(\mathbf{s}) \\
&< p/2
\end{aligned}$$

Furthermore, modulo r , we have $\llbracket \langle \mathbf{c}', \mathbf{s} \rangle \rrbracket_p = e_p = \langle \mathbf{c}', \mathbf{s} \rangle - kp = \langle \mathbf{c}, \mathbf{s} \rangle - kq = \llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q$. □

The lemma implies that an evaluator, who does not know the secret key but instead only knows a bound on its length, can potentially transform a ciphertext \mathbf{c} that encrypts m under key \mathbf{s} for modulus q – i.e., $m = [[\langle \mathbf{c}, \mathbf{s} \rangle]_q]_r$ – into a ciphertext \mathbf{c}' that encrypts m under the same key \mathbf{s} for modulus p – i.e., $m = [[\langle \mathbf{c}', \mathbf{s} \rangle]_p]_r$. Specifically, the following corollary follows immediately from Lemma 4.

Corollary 1. *Let p and q be two odd moduli. Suppose \mathbf{c} is an encryption of bit m under key \mathbf{s} for modulus q – i.e., $m = [[\langle \mathbf{c}, \mathbf{s} \rangle]_q]_r$. Moreover, suppose that \mathbf{s} is a fairly short key and the “noise” $e_q \leftarrow [\langle \mathbf{c}, \mathbf{s} \rangle]_q$ has small magnitude – precisely, assume that $\|e_q\| < q/2 - (q/p) \cdot (r/2) \cdot \sqrt{d} \cdot \gamma(R) \cdot \ell_1^{(R)}(\mathbf{s})$. Then $\mathbf{c}' \leftarrow \text{Scale}(\mathbf{c}, q, p, r)$ is an encryption of bit m under key \mathbf{s} for modulus p – i.e., $m = [[\langle \mathbf{c}', \mathbf{s} \rangle]_p]_r$. The noise $e_p = [\langle \mathbf{c}', \mathbf{s} \rangle]_p$ of the new ciphertext has magnitude at most $(p/q) \cdot \|\langle \mathbf{c}, \mathbf{s} \rangle\| + \gamma(R) \cdot (r/2) \cdot \sqrt{d} \cdot \ell_1^{(R)}(\mathbf{s})$.*

Amazingly, assuming p is smaller than q and \mathbf{s} has coefficients that are small in relation to q , this trick permits the evaluator to reduce the magnitude of the noise without knowing the secret key! (Of course, this is also what Gentry’s bootstrapping transformation accomplishes, but in a much more complicated way.)

3.4 (Leveled) FHE Based on GLWE without Bootstrapping

We now present our FHE scheme. Given the machinery that we have described in the previous subsections, the scheme itself is remarkably simple.

In our scheme, we will use a parameter L indicating the number of levels of arithmetic circuit that we want our FHE scheme to be capable of evaluating. Note that this is an exponential improvement over prior schemes, that would typically use a parameter d indicating the *degree* of the polynomials to be evaluated.

(Note: the linear polynomial L^{long} , used below, is defined in Section 3.2.)

Our FHE Scheme without Bootstrapping:

- $\text{FHE.Setup}(1^\lambda, 1^L, b)$: Takes as input the security parameter, a number of levels L , and a bit b . Use the bit $b \in \{0, 1\}$ to determine whether we are setting parameters for a LWE-based scheme (where $d = 1$) or a RLWE-based scheme (where $n = 1$). Let $\mu = \mu(\lambda, L, b) = \theta(\log \lambda + \log L)$ be a parameter that we will specify in detail later. For $j = L$ (input level of circuit) to 0 (output level), run $\text{params}_j \leftarrow \text{E.Setup}(1^\lambda, 1^{(j+1) \cdot \mu}, b)$ to obtain a ladder of decreasing moduli from $q_L ((L + 1) \cdot \mu$ bits) down to q_0 (μ bits). For $j = L - 1$ to 0, replace the value of d_j in params_j with $d = d_L$ and the distribution χ_j with $\chi = \chi_L$. (That is, the ring dimension and noise distribution do not depend on the circuit level, but the vector dimension n_j still might.)
- $\text{FHE.KeyGen}(\{\text{params}_j\})$: For $j = L$ down to 0, do the following:
 1. Run $\mathbf{s}_j \leftarrow \text{E.SecretKeyGen}(\text{params}_j)$ and $\mathbf{A}_j \leftarrow \text{E.PublicKeyGen}(\text{params}_j, \mathbf{s}_j)$.
 2. Set $\mathbf{s}'_j \leftarrow \mathbf{s}_j \otimes \mathbf{s}_j \in R_{q_j}^{\binom{n_j+1}{2}}$. That is, \mathbf{s}'_j is a tensoring of \mathbf{s}_j with itself whose coefficients are each the product of two coefficients of \mathbf{s}_j in R_{q_j} .
 3. Set $\mathbf{s}''_j \leftarrow \text{BitDecomp}(\mathbf{s}'_j, q_j)$.
 4. Run $\tau_{\mathbf{s}''_{j+1} \rightarrow \mathbf{s}_j} \leftarrow \text{SwitchKeyGen}(\mathbf{s}''_j, \mathbf{s}_{j-1})$. (Omit this step when $j = L$.)

The secret key sk consists of the \mathbf{s}_j ’s and the public key pk consists of the \mathbf{A}_j ’s and $\tau_{\mathbf{s}''_{j+1} \rightarrow \mathbf{s}_j}$ ’s.

- $\text{FHE.Enc}(\text{params}, pk, m)$: Take a message in R_2 . Run $\text{E.Enc}(\mathbf{A}_L, m)$.
- $\text{FHE.Dec}(\text{params}, sk, \mathbf{c})$: Suppose the ciphertext is under key \mathbf{s}_j . Run $\text{E.Dec}(\mathbf{s}_j, \mathbf{c})$. (The ciphertext could be augmented with an index indicating which level it belongs to.)

- $\text{FHE.Add}(pk, c_1, c_2)$: Takes two ciphertexts encrypted under the same s_j . (If they are not initially, use FHE.Refresh (below) to make it so.) Set $c_3 \leftarrow c_1 + c_2 \bmod q_j$. Interpret c_3 as a ciphertext under s'_j (s'_j 's coefficients include all of s_j 's since $s'_j = s_j \otimes s_j$ and s_j 's first coefficient is 1) and output:

$$c_4 \leftarrow \text{FHE.Refresh}(c_3, \tau_{s''_j \rightarrow s_{j-1}}, q_j, q_{j-1})$$

- $\text{FHE.Mult}(pk, c_1, c_2)$: Takes two ciphertexts encrypted under the same s_j . If they are not initially, use FHE.Refresh (below) to make it so.) First, multiply: the new ciphertext, under the secret key $s'_j = s_j \otimes s_j$, is the coefficient vector c_3 of the linear equation $L_{c_1, c_2}^{long}(x \otimes x)$. Then, output:

$$c_4 \leftarrow \text{FHE.Refresh}(c_3, \tau_{s''_j \rightarrow s_{j-1}}, q_j, q_{j-1})$$

- $\text{FHE.Refresh}(c, \tau_{s''_j \rightarrow s_{j-1}}, q_j, q_{j-1})$: Takes a ciphertext encrypted under s'_j , the auxiliary information $\tau_{s''_j \rightarrow s_{j-1}}$ to facilitate key switching, and the current and next moduli q_j and q_{j-1} . Do the following:

1. Expand: Set $c_1 \leftarrow \text{Powersof2}(c, q_j)$. (Observe: $\langle c_1, s''_j \rangle = \langle c, s'_j \rangle \bmod q_j$ by Lemma 2.)
2. Switch Moduli: Set $c_2 \leftarrow \text{Scale}(c_1, q_j, q_{j-1}, 2)$, a ciphertext under the key s''_j for modulus q_{j-1} .
3. Switch Keys: Output $c_3 \leftarrow \text{SwitchKey}(\tau_{s''_j \rightarrow s_{j-1}}, c_2, q_{j-1})$, a ciphertext under the key s_{j-1} for modulus q_{j-1} .

Remark 1. We mention the obvious fact that, since addition increases the noise much more slowly than multiplication, one does not necessarily need to refresh after additions, even high fan-in ones.

The key step of our new FHE scheme is the Refresh procedure. If the modulus q_{j-1} is chosen to be smaller than q_j by a sufficient multiplicative factor, then Corollary 1 implies that the noise of the ciphertext output by Refresh is smaller than that of the input ciphertext – that is, the ciphertext will indeed be a “refreshed” encryption of the same value. We elaborate on this analysis in the next section.

One can reasonably argue that this scheme is not “FHE without bootstrapping” since $\tau_{s''_j \rightarrow s_{j-1}}$ can be viewed as an encrypted secret key, and the SwitchKey step can be viewed as a homomorphic evaluation of the decryption function. We prefer not to view the SwitchKey step this way. While there is some high-level resemblance, the low-level details are very different, a difference that becomes tangible in the much better asymptotic performance. To the extent that it performs decryption, SwitchKey does so very efficiently using an efficient (not bit-wise) representation of the secret key that allows this step to be computed in quasi-linear time for the RLWE instantiation, below the quadratic lower bound for bootstrapping. Certainly SwitchKey does not use the usual ponderous approach of representing the decryption function as a boolean circuit to be traversed homomorphically. Another difference is that the SwitchKey step does not actually reduce the noise level (as bootstrapping does); rather, the noise is reduced by the Scale step.

4 Correctness, Setting the Parameters, Performance, and Security

Here, we will show how to set the parameters of the scheme so that the scheme is correct. Mostly, this involves analyzing each of the steps within FHE.Add and FHE.Mult – namely, the addition or multiplication itself, and then the Powersof2, Scale and SwitchKey steps that make up FHE.Refresh – to establish that the output of each step is a decryptable ciphertext with bounded noise. This analysis will lead to concrete suggestions for how to set the ladder of moduli and to asymptotic bounds on the performance of the scheme.

Let us begin by considering how much noise FHE.Enc introduces initially.

4.1 The Initial Noise from FHE.Enc

Recall that FHE.Enc simply invokes E.Enc for suitable parameters ($params_L$) that depend on λ and L . In turn, the noise of ciphertexts output by E.Enc depends on the noise of the initial “ciphertexts” (the encryptions of 0) implicit in the matrix A output by E.PublicKeyGen, whose noise distribution is dictated by the distribution χ .

Lemma 5. *Let n_L and q_L be the parameters associated to FHE.Enc. Let d be the dimension of the ring R , and let γ_R be the expansion factor associated to R . (Both of these quantities are 1 when $R = \mathbb{Z}$.) Let B_χ be a bound such that R -elements sampled from the noise distribution χ have length at most B_χ with overwhelming probability. The length of the noise in ciphertexts output by FHE.Enc is at most $1 + 2 \cdot \gamma_R \cdot \sqrt{d} \cdot ((2n_L + 1) \log q_L) \cdot B_\chi$.*

Proof. Recall that $\mathbf{s} \leftarrow \text{E.SecretKeyGen}$ and $\mathbf{A} \leftarrow \text{E.PublicKeyGen}(\mathbf{s}, N)$ for $N = (2n_L + 1) \log q_L$, where $\mathbf{A} \cdot \mathbf{s} = 2\mathbf{e}$ for $\mathbf{e} \leftarrow \chi$. Recall that encryption works as follows: $\mathbf{c} \leftarrow \mathbf{m} + \mathbf{A}^T \mathbf{r} \bmod q$ where $\mathbf{r} \in R_2^N$. We have that the noise of this ciphertext is $[\langle \mathbf{c}, \mathbf{s} \rangle]_q = [m + 2\langle \mathbf{r}, \mathbf{e} \rangle]_q$, whose magnitude is at most $1 + 2 \cdot \gamma_R \cdot \sum_{j=1}^N \|\mathbf{r}[j]\| \cdot \|\mathbf{e}[j]\| \leq 1 + 2 \cdot \gamma_R \cdot \sqrt{d} \cdot N \cdot B_\chi$. \square

Notice that we are using very loose (i.e., conservative) upper bounds for the noise. These bounds could be tightened up with a more careful analysis. The correctness of decryption for ciphertexts output by FHE.Enc, assuming the noise bound above is less than $q/2$, follows directly from the correctness of the basic encryption and decryption algorithms E.Enc and E.Dec.

4.2 Correctness and Performance of FHE.Add and FHE.Mult (before FHE.Refresh)

Consider FHE.Mult. One begins $\text{FHE.Mult}(pk, \mathbf{c}_1, \mathbf{c}_2)$ with two ciphertexts under key \mathbf{s}_j for modulus q_j that have noises $e_i = [L_{\mathbf{c}_i}(\mathbf{s}_j)]_{q_j}$, where $L_{\mathbf{c}_i}(\mathbf{x})$ is simply the dot product $\langle \mathbf{c}_i, \mathbf{x} \rangle$. To multiply together two ciphertexts, one multiplies together these two linear equations to obtain a quadratic equation $Q_{\mathbf{c}_1, \mathbf{c}_2}(\mathbf{x}) \leftarrow L_{\mathbf{c}_1}(\mathbf{x}) \cdot L_{\mathbf{c}_2}(\mathbf{x})$, and then interprets this quadratic equation as a linear equation $L_{\mathbf{c}_1, \mathbf{c}_2}^{\text{long}}(\mathbf{x} \otimes \mathbf{x}) = Q_{\mathbf{c}_1, \mathbf{c}_2}(\mathbf{x})$ over the tensored vector $\mathbf{x} \otimes \mathbf{x}$. The coefficients of this long linear equation compose the new ciphertext vector \mathbf{c}_3 . Clearly, $[\langle \mathbf{c}_3, \mathbf{s}_j \otimes \mathbf{s}_j \rangle]_{q_j} = [L_{\mathbf{c}_1, \mathbf{c}_2}^{\text{long}}(\mathbf{s}_j \otimes \mathbf{s}_j)]_{q_j} = [e_1 \cdot e_2]_{q_j}$. Thus, if the noises of \mathbf{c}_1 and \mathbf{c}_2 have length at most B , then the noise of \mathbf{c}_3 has length at most $\gamma_R \cdot B^2$, where γ_R is the expansion factor of R . If this length is less than $q_j/2$, then decryption works correctly. In particular, if $m_i = [\langle \mathbf{c}_i, \mathbf{s}_j \rangle]_{q_j} \bmod 2 = [e_i]_2$ for $i \in \{1, 2\}$, then over R_2 we have $[\langle \mathbf{c}_3, \mathbf{s}_j \otimes \mathbf{s}_j \rangle]_{q_j} \bmod 2 = [[e_1 \cdot e_2]_{q_j}]_2 = [e_1 \cdot e_2]_2 = [e_1]_2 \cdot [e_2]_2 = m_1 \cdot m_2$. That is, correctness is preserved as long as this noise does not wrap modulo q_j .

The correctness of FHE.Add and FHE.Mult (before FHE.Refresh) is formally captured in the following lemmas.

Lemma 6. *Let \mathbf{c}_1 and \mathbf{c}_2 be two ciphertexts under key \mathbf{s}_j for modulus q_j , where $\|[\langle \mathbf{c}_i, \mathbf{s}_j \rangle]_{q_j}\| \leq B$ and $m_i = [[\langle \mathbf{c}_i, \mathbf{s}_j \rangle]_{q_j}]_2$. Let $\mathbf{s}'_j = \mathbf{s}_j \otimes \mathbf{s}_j$, where the “non-quadratic coefficients” of \mathbf{s}'_j (namely, the ‘1’ and the coefficients of \mathbf{s}_j) are placed first. Let $\mathbf{c}' = \mathbf{c}_1 + \mathbf{c}_2$, and pad \mathbf{c}' with zeros to get a vector \mathbf{c}_3 such that $\langle \mathbf{c}_3, \mathbf{s}'_j \rangle = \langle \mathbf{c}', \mathbf{s}_j \rangle$. The noise $[\langle \mathbf{c}_3, \mathbf{s}'_j \rangle]_{q_j}$ has length at most $2B$. If $2B < q_j/2$, \mathbf{c}_3 is an encryption of $m_1 + m_2$ under key \mathbf{s}'_j for modulus q_j – i.e., $m_1 \cdot m_2 = [[\langle \mathbf{c}_3, \mathbf{s}'_j \rangle]_{q_j}]_2$.*

Lemma 7. *Let \mathbf{c}_1 and \mathbf{c}_2 be two ciphertexts under key \mathbf{s}_j for modulus q_j , where $\|[\langle \mathbf{c}_i, \mathbf{s}_j \rangle]_{q_j}\| \leq B$ and $m_i = [[\langle \mathbf{c}_i, \mathbf{s}_j \rangle]_{q_j}]_2$. Let the linear equation $L_{\mathbf{c}_1, \mathbf{c}_2}^{\text{long}}(\mathbf{x} \otimes \mathbf{x})$ be as defined above, let \mathbf{c}_3 be the coefficient vector of this linear equation, and let $\mathbf{s}'_j = \mathbf{s}_j \otimes \mathbf{s}_j$. The noise $[\langle \mathbf{c}_3, \mathbf{s}'_j \rangle]_{q_j}$ has length at most $\gamma_R \cdot B^2$. If $\gamma_R \cdot B^2 < q_j/2$, \mathbf{c}_3 is an encryption of $m_1 \cdot m_2$ under key \mathbf{s}'_j for modulus q_j – i.e., $m_1 \cdot m_2 = [[\langle \mathbf{c}_3, \mathbf{s}'_j \rangle]_{q_j}]_2$.*

The computation needed to compute the tensored ciphertext c_3 is $\tilde{O}(dn_j^2 \log q_j)$. For the RLWE instantiation, since $n_j = 1$ and since (as we will see) $\log q_j$ depends logarithmically on the security parameter and linearly on L , the computation here is only quasi-linear in the security parameter. For the LWE instantiation, the computation is quasi-quadratic.

4.3 Correctness and Performance of FHE.Refresh

FHE.Refresh consists of three steps: Expand, Switch Moduli, and Switch Keys. We address each of these steps in turn.

Correctness and Performance of the Expand Step. The Expand step of FHE.Refresh takes as input a long ciphertext c under the long tensored key $s'_j = s_j \otimes s_j$ for modulus q_j . It simply applies the Powersof2 transformation to c to obtain c_1 . By Lemma 2, we know that

$$\langle \text{Powersof2}(c, q_j), \text{BitDecomp}(s'_j, q_j) \rangle = \langle c, s'_j \rangle \bmod q_j$$

i.e., we know that if s'_j decrypts c correctly, then s''_j decrypts c_1 correctly. The noise has not been affected at all.

If implemented naively, the computation in the Expand step is $\tilde{O}(dn_j^2 \log^2 q_j)$. The somewhat high computation is due to the fact that the expanded ciphertext is a $((\binom{n_j+1}{2}) \cdot \lceil \log q_j \rceil)$ -dimensional vector over R_q .

However, recall that s_j is drawn using the distribution χ – i.e., it has small coefficients of size basically independent of q_j . Consequently, s'_j also has small coefficients, and we can use this *a priori* knowledge in combination with an optimized version of BitDecomp to output a shorter bit decomposition of s'_j – in particular, a $((\binom{n_j+1}{2}) \cdot \lceil \log q'_j \rceil)$ -dimensional vector over R_q where $q'_j \ll q_j$ is a bound (with overwhelming probability) on the coefficients of elements output by χ . Similarly, we can use an abbreviated version of Powersof2(c, q_j). In this case, the computation is $\tilde{O}(dn_j^2 \log q_j)$.

Correctness and Performance of the Switch-Moduli Step. The Switch Moduli step takes as input a ciphertext c_1 under the secret bit-vector s''_j for the modulus q_j , and outputs the ciphertext $c_2 \leftarrow \text{Scale}(c_1, q_j, q_{j-1}, 2)$, which we claim to be a ciphertext under key s''_j for modulus q_{j-1} . Note that s''_j is a *short* secret key, since it is a bit vector in $R_2^{t_j}$ for $t_j \leq (\binom{n_j+1}{2}) \cdot \lceil \log q_j \rceil$. By Corollary 1, and using the fact that $\ell_1(s''_j) \leq \sqrt{d} \cdot t_j$, the following is true: if the noise of c_1 has length at most $B < q_j/2 - (q_j/q_{j-1}) \cdot d \cdot \gamma_R \cdot t_j$, then correctness is preserved and the noise of c_2 is bounded by $(q_{j-1}/q_j) \cdot B + d \cdot \gamma_R \cdot t_j$. Of course, the key feature of this step for our purposes is that switching moduli may *reduce* the length of the moduli when $q_{j-1} < q_j$.

We capture the correctness of the Switch-Moduli step in the following lemma.

Lemma 8. *Let c_1 be a ciphertext under the key $s''_j = \text{BitDecomp}(s_j \otimes s_j, q_j)$ such that $e_j \leftarrow [\langle c_1, s''_j \rangle]_{q_j}$ has length at most B and $m = [e_j]_2$. Let $c_2 \leftarrow \text{Scale}(c_1, q_j, q_{j-1}, 2)$, and let $e_{j-1} = [\langle c_2, s''_j \rangle]_{q_{j-1}}$. Then, e_{j-1} (the new noise) has length at most $(q_{j-1}/q_j) \cdot B + d \cdot \gamma_R \cdot (\binom{n_j+1}{2}) \cdot \lceil \log q_j \rceil$, and (assuming this noise length is less than $q_{j-1}/2$) we have $m = [e_{j-1}]_2$.*

The computation in the Switch-Moduli step is $\tilde{O}(dn_j^2 \log q_j)$, using the optimized versions of BitDecomp and Powersof2 mentioned above.

Correctness and Performance of the Switch-Key Step. Finally, in the Switch Keys step, we take as input a ciphertext c_2 under key s''_j for modulus q_{j-1} and set $c_3 \leftarrow \text{SwitchKey}(\tau_{s''_j \rightarrow s_{j-1}}, c_2, q_{j-1})$, a ciphertext under the key s_{j-1} for modulus q_{j-1} . In Lemma 3, we proved the correctness of key switching and established that the noise grows only by the additive factor $2 \langle \text{BitDecomp}(c_2, q_{j-1}), e \rangle$, where $\text{BitDecomp}(c_2, q_{j-1})$ is

a (short) bit-vector and \mathbf{e} is a (short and fresh) noise vector. In particular, if the noise originally had length B , then after the Switch Keys step is has length at most $B + 2 \cdot \gamma_R \cdot \sum_{i=1}^{u_j} \|\text{BitDecomp}(\mathbf{c}_2, q_{j-1})[i]\| \cdot B_\chi \leq B + 2 \cdot \gamma_R \cdot u_j \cdot \sqrt{d} \cdot B_\chi$, where $u_j \leq \binom{n_j+1}{2} \cdot \lceil \log q_j \rceil \cdot \lceil \log q_{j-1} \rceil$ is the dimension of $\text{BitDecomp}(\mathbf{c}_2)$.

We capture the correctness of the Switch-Key step in the following lemma.

Lemma 9. *Let \mathbf{c}_2 be a ciphertext under the key $\mathbf{s}_j'' = \text{BitDecomp}(\mathbf{s}_j \otimes \mathbf{s}_j, q_j)$ for modulus q_{j-1} such that $e_1 \leftarrow [\langle \mathbf{c}_2, \mathbf{s}_j'' \rangle]_{q_{j-1}}$ has length at most B and $m = [e_1]_2$. Let $\mathbf{c}_3 \leftarrow \text{SwitchKey}(\tau_{\mathbf{s}_j'' \rightarrow \mathbf{s}_{j-1}}, \mathbf{c}_2, q_{j-1})$, and let $e_2 = [\langle \mathbf{c}_3, \mathbf{s}_{j-1} \rangle]_{q_{j-1}}$. Then, e_2 (the new noise) has length at most $B + 2 \cdot \gamma_R \cdot \binom{n_j+1}{2} \cdot \lceil \log q_j \rceil^2 \cdot \sqrt{d} \cdot B_\chi$ and (assuming this noise length is less than $q_{j-1}/2$) we have $m = [e_2]_2$.*

In terms of computation, the Switch-Key step involves multiplying the transpose of u_j -dimensional vector $\text{BitDecomp}(\mathbf{c}_2)$ with a $u_j \times (n_{j-1} + 1)$ matrix B . Assuming $n_j \geq n_{j-1}$ and $q_j \geq q_{j-1}$, and using the optimized versions of BitDecomp and Powersof2 mentioned above to reduce u_j , this computation is $\tilde{O}(dn_j^3 \log^2 q_j)$. Still this is quasi-linear in the RLWE instantiation.

4.4 Putting the Pieces Together: Parameters, Correctness, Performance

So far we have established that the scheme is correct, *assuming* that the noise does not wrap modulo q_j or q_{j-1} . Now we need to show that we can set the parameters of the scheme to ensure that such wrapping never occurs.

Our strategy for setting the parameters is to pick a “universal” bound B on the noise length, and then prove, for all j , that a valid ciphertext under key \mathbf{s}_j for modulus q_j has noise length at most B . This bound B is quite small: polynomial in λ and $\log q_L$, where q_L is the largest modulus in our ladder. It is clear that such a bound B holds for fresh ciphertexts output by FHE.Enc . (Recall the discussion from Section 3.1 where we explained that we use a noise distribution χ that is essentially independent of the modulus.) The remainder of the proof is by induction – i.e., we will show that if the bound holds for two ciphertexts $\mathbf{c}_1, \mathbf{c}_2$ at level j , our lemmas above imply that the bound also holds for the ciphertext $\mathbf{c}' \leftarrow \text{FHE.Mult}(pk, \mathbf{c}_1, \mathbf{c}_2)$ at level $j - 1$. (FHE.Mult increases the noise strictly more in the worst-case than FHE.Add for any reasonable choice of parameters.)

Specifically, after the first step of FHE.Mult (without the Refresh step), the noise has length at most $\gamma_R \cdot B^2$. Then, we apply the Scale function, after which the noise length is at most $(q_{j-1}/q_j) \cdot \gamma_R \cdot B^2 + \eta_{\text{Scale},j}$, where $\eta_{\text{Scale},j}$ is some additive term. Finally, we apply the SwitchKey function, which introduces another additive term $\eta_{\text{SwitchKey},j}$. Overall, after the entire FHE.Mult step, the noise length is at most $(q_{j-1}/q_j) \cdot \gamma_R \cdot B^2 + \eta_{\text{Scale},j} + \eta_{\text{SwitchKey},j}$. We want to choose our parameters so that this bound is at most B . Suppose we set our ladder of moduli and the bound B such that the following two properties hold:

- Property 1: $B \geq 2 \cdot (\eta_{\text{Scale},j} + \eta_{\text{SwitchKey},j})$ for all j .
- Property 2: $q_j/q_{j-1} \geq 2 \cdot B \cdot \gamma_R$ for all j .

Then we have

$$(q_{j-1}/q_j) \cdot \gamma_R \cdot B^2 + \eta_{\text{Scale},j} + \eta_{\text{SwitchKey},j} \leq \frac{1}{2 \cdot B \cdot \gamma_R} \cdot \gamma_R \cdot B^2 + \frac{1}{2} \cdot B \leq B$$

It only remains to set our ladder of moduli and B so that Properties 1 and 2 hold.

Unfortunately, there is some circularity in Properties 1 and 2: q_L depends on B , which depends on q_L , albeit only polylogarithmically. However, it is easy to see that this circularity is not fatal. As a non-optimized example to illustrate this, set $B = \lambda^a \cdot L^b$ for very large constants a and b , and set $q_j \approx 2^{(j+1) \cdot \omega(\log \lambda + \log L)}$.

If a and b are large enough, B dominates $\eta_{\text{Scale},L} + \eta_{\text{SwitchKey},L}$, which is polynomial in λ and $\log q_L$, and hence polynomial in λ and L (Property 1 is satisfied). Since q_j/q_{j-1} is super-polynomial in both λ and L , it dominates $2 \cdot B \cdot \gamma_R$ (Property 2 is satisfied). In fact, it works fine to set q_j as a modulus having $(j+1) \cdot \mu$ bits for some $\mu = \theta(\log \lambda + \log L)$ with small hidden constant.

Overall, we have that q_L , the largest modulus used in the system, is $\theta(L \cdot (\log \lambda + \log L))$ bits, and $d \cdot n_L$ must be approximately that number times λ for 2^λ security.

Theorem 3. *For some $\mu = \theta(\log \lambda + \log L)$, FHE is a correct L -leveled FHE scheme – specifically, it correctly evaluates circuits of depth L with Add and Mult gates over R_2 . The per-gate computation is $\tilde{O}(d \cdot n_L^3 \cdot \log^2 q_j) = \tilde{O}(d \cdot n_L^3 \cdot L^2)$. For the LWE case (where $d = 1$), the per-gate computation is $\tilde{O}(\lambda^3 \cdot L^5)$. For the RLWE case (where $n = 1$), the per-gate computation is $\tilde{O}(\lambda \cdot L^3)$.*

The bottom line is that we have a RLWE-based leveled FHE scheme with per-gate computation that is only *quasi-linear* in the security parameter, albeit with somewhat high dependence on the number of levels in the circuit.

Let us pause at this point to reconsider the performance of previous FHE schemes in comparison to our new scheme. Specifically, as we discussed in the Introduction, in previous SWHE schemes, the ciphertext size is at least $\tilde{O}(\lambda \cdot d^2)$, where d is the *degree* of the circuit being evaluated. One may view our new scheme as a very powerful SWHE scheme in which this dependence on *degree* has been replaced with a similar dependence on *depth*. (Recall the degree of a circuit may be exponential in its depth.) Since polynomial-size circuits have polynomial depth, which is certainly not true of *degree*, our scheme can efficiently evaluate arbitrary circuits without resorting to bootstrapping.

4.5 Security

The security of FHE follows by a standard hybrid argument from the security of E, the basic scheme described in Section 3.1. We omit the details.

5 Optimizations

Despite the fact that our new FHE scheme has per-gate computation only quasi-linear in the security parameter, we present several significant ways of optimizing it. We focus primarily on the RLWE-based scheme, since it is much more efficient.

Our first optimization is *batching*. Batching allows us to reduce the per-gate computation from quasi-linear in the security parameter to *polylogarithmic*. In more detail, we show that evaluating a function f homomorphically on $\ell = \Omega(\lambda)$ blocks of encrypted data requires only *polylogarithmically* (in terms of the security parameter λ) more computation than evaluating f on the unencrypted data. (The overhead is still polynomial in the depth L of the circuit computing f .) Batching works essentially by packing multiple plaintexts into each ciphertext.

Next, we reintroduce *bootstrapping* as an optimization rather than a necessity (Section 5.2). Bootstrapping allows us to achieve per-gate computation *quasi-quadratic* in the security parameter, *independent* of the number levels in the circuit being evaluated.

In Section 5.3, we show that *batching the bootstrapping function* is a powerful combination. With this optimization, circuits whose levels mostly have width at least λ can be evaluated homomorphically with only $\tilde{O}(\lambda)$ per-gate computation, independent of the number of levels.

Finally, Section 5.5 presents a few other miscellaneous optimizations.

5.1 Batching

Suppose we want to evaluate the same function f on ℓ blocks of encrypted data. (Or, similarly, suppose we want to evaluate the same encrypted function f on ℓ blocks of plaintext data.) Can we do this using less than

ℓ times the computation needed to evaluate f on one block of data? Can we batch?

For example, consider a keyword search function that returns ‘1’ if the keyword is present in the data and ‘0’ if it is not. The keyword search function is mostly composed of a large number of equality tests that compare the target word w to all of the different subsequences of data; this is followed up by an OR of the equality test results. All of these equality tests involve running the same w -dependent function on different blocks of data. If we could batch these equality tests, it could significantly reduce the computation needed to perform keyword search homomorphically.

If we use bootstrapping as an optimization (see Section 5.2), then obviously we will be running the decryption function homomorphically on multiple blocks of data – namely, the multiple ciphertexts that need to be refreshed. Can we batch the bootstrapping function? If we could, then we might be able to drastically reduce the average per-gate cost of bootstrapping.

Smart and Vercauteren [21] were the first to rigorously analyze batching in the context of FHE. In particular, they observed that ideal-lattice-based (and RLWE-based) ciphertexts can have many plaintext slots, associated to the factorization of the plaintext space into algebraic ideals.

When we apply batching to our new RLWE-based FHE scheme, the results are pretty amazing. Evaluating f homomorphically on $\ell = \Omega(\lambda)$ blocks of encrypted data requires only *polylogarithmically* (in terms of the security parameter λ) more computation than evaluating f on the unencrypted data. (The overhead is still polynomial in the depth L of the circuit computing f .) As we will see later, for circuits whose levels mostly have width at least λ , *batching the bootstrapping function* (i.e., batching homomorphic evaluation of the decryption function) allows us to reduce the per-gate computation of our bootstrapped scheme from $\tilde{O}(\lambda^2)$ to $\tilde{O}(\lambda)$ (independent of L).

To make the exposition a bit simpler, in our RLWE-based instantiation where $R = \mathbb{Z}[x]/(x^d + 1)$, we will not use R_2 as our plaintext space, but instead use a plaintext space R_p that is isomorphic to the direct product $R_{p_1} \times \cdots \times R_{p_d}$ of many plaintext spaces (think Chinese remaindering), so that evaluating a function *once* over R_p implicitly evaluates the function *many* times in parallel over the respective smaller plaintext spaces. The \mathfrak{p}_i ’s will be *ideals* in our ring $R = \mathbb{Z}[x]/(x^d + 1)$. (One could still use R_2 as in [21], but the number theory there is a bit more involved.)

5.1.1 Some Number Theory

Let us take a very brief tour of algebraic number theory. Suppose p is a prime number satisfying $p \equiv 1 \pmod{2d}$, and let a be a primitive $2d$ -th root of unity modulo p . Then, $x^d + 1$ factors completely into linear polynomials modulo p – in particular, $x^d + 1 = \prod_{i=1}^d (x - a_i) \pmod{p}$ where $a_i = a^{2i-1} \pmod{p}$. In some sense, the converse of the above statement is also true, and this is the essence of *reciprocity* – namely, in the ring $R = \mathbb{Z}[x]/(x^d + 1)$ the prime integer p is not actually prime, but rather it splits completely into prime ideals in R – i.e., $p = \prod_{i=1}^d \mathfrak{p}_i$. The ideal \mathfrak{p}_i equals $(p, x - a_i)$ – namely, the set of all R -elements that can be expressed as $r_1 \cdot p + r_2 \cdot (x - a_i)$ for some $r_1, r_2 \in R$. Each ideal \mathfrak{p}_i has norm p – that is, roughly speaking, a $1/p$ fraction of R -elements are in \mathfrak{p}_i , or, more formally, the p cosets $0 + \mathfrak{p}_i, \dots, (p-1) + \mathfrak{p}_i$ partition R . These ideals are relative prime, and so they behave like relative prime integers. In particular, the Chinese Remainder Theorem applies: $R_p \cong R_{\mathfrak{p}_1} \times \cdots \times R_{\mathfrak{p}_d}$.

Although the prime ideals $\{\mathfrak{p}_i\}$ are relatively prime, they are close siblings, and it is easy, in some sense, to switch from one to another. One fact that we will use (when we finally apply batching to bootstrapping) is that, for any i, j there is an automorphism $\sigma_{i \rightarrow j}$ over R that maps elements of \mathfrak{p}_i to elements of \mathfrak{p}_j . Specifically, $\sigma_{i \rightarrow j}$ works by mapping an R -element $r = r(x) = r_{d-1}x^{d-1} + \cdots + r_1x + r_0$ to $r(x^{e_{ij}}) = r_{d-1}x^{e_{ij}(d-1) \bmod 2d} + \cdots + r_1x^{e_{ij}} + r_0$ where e_{ij} is some odd number in $[1, 2d]$. Notice that this automorphism just permutes the coefficients of r and fixes the free coefficient. Notationally, we will use $\sigma_{i \rightarrow j}(\mathbf{v})$ to refer to the vector that results from applying $\sigma_{i \rightarrow j}$ coefficient-wise to \mathbf{v} .

5.1.2 How Batching Works

Deploying batching inside our scheme FHE is quite straightforward. First, we pick a prime $p = 1 \bmod 2d$ of size polynomial in the security parameter. (One should exist under the GRH.)

The next step is simply to recognize that our scheme FHE works just fine when we replace the original plaintext space R_2 with R_p . There is nothing especially magical about the number 2. In the basic scheme E described in Section 3.1, $\text{E.PublicKeyGen}(params, sk)$ is modified in the obvious way so that $\mathbf{A} \cdot \mathbf{s} = p \cdot \mathbf{e}$ rather than $2 \cdot \mathbf{e}$. (This modification induces a similar modification in SwitchKeyGen .) Decryption becomes $m = \llbracket \langle \mathbf{c}, \mathbf{s} \rangle \rrbracket_q \pmod p$. Homomorphic operations use mod- p gates rather than boolean gates, and it is easy (if desired) to emulate boolean gates with mod- p gates – e.g., we can compute $\text{XOR}(a, b)$ for $a, b \in \{0, 1\}^2$ using mod- p gates for any p as $a + b - 2ab$. For modulus switching, we use $\text{Scale}(\mathbf{c}_1, q_j, q_{j-1}, p)$ rather than $\text{Scale}(\mathbf{c}_1, q_j, q_{j-1}, 2)$. The larger rounding error from this new scaling procedure increases the noise slightly, but this additive noise is still polynomial in the security parameter and the number of levels, and thus is still consistent with our setting of parameters. In short, FHE can easily be adapted to work with a plaintext space R_p for p of polynomial size.

The final step is simply to recognize that, by the Chinese Remainder Theorem, evaluating an arithmetic circuit over R_p on input $\mathbf{x} \in R_p^n$ implicitly evaluates, for each i , the same arithmetic circuit over R_{p_i} on input \mathbf{x} projected down to $R_{p_i}^n$. The evaluations modulo the various prime ideals do not “mix” or interact with each other.

Theorem 4. *Let $p = 1 \bmod 2d$ be a prime of size polynomial in λ . The RLWE-based instantiation of FHE using the ring $R = \mathbb{Z}[x]/(x^d + 1)$ can be adapted to use the plaintext space $R_p = \bigotimes_{i=1}^d R_{p_i}$ while preserving correctness and the same asymptotic performance. For any boolean circuit f of depth L , the scheme can homomorphically evaluate f on ℓ sets of inputs with per-gate computation $\tilde{O}(\lambda \cdot L^3 / \min\{d, \ell\})$.*

When $\ell \geq \lambda$, the per-gate computation is only polylogarithmic in the security parameter (still cubic in L).

5.2 Bootstrapping as an Optimization

Bootstrapping is no longer strictly necessary to achieve leveled FHE. However, in some settings, it may have some advantages:

- **Performance:** The per-gate computation is independent of the depth of the circuit being evaluated.
- **Flexibility:** Assuming circular security, a bootstrapped scheme can perform homomorphic evaluations indefinitely without needing to specify in advance, during Setup, a bound on the number of circuit levels.
- **Memory:** Bootstrapping permits short ciphertexts – e.g., encrypted using AES – to be de-compressed to longer ciphertexts that permit homomorphic operations. Bootstrapping allows us to save memory by storing data encrypted in the compressed form – e.g., under AES.

Here, we revisit bootstrapping, viewing it as an optimization rather than a necessity. We also reconsider the scheme FHE that we described in Section 3, viewing the scheme not as an end in itself, but rather as a very powerful SWHE whose performance degrades polynomially in the *depth* of the circuit being evaluated, as opposed to previous SWHE schemes whose performance degrades polynomially in the *degree*. In particular, we analyze how efficiently it can evaluate its decryption function, as needed to bootstrap. Not surprisingly, our faster SWHE scheme can also bootstrap faster. The decryption function has only logarithmic depth and can be evaluated homomorphically in time quasi-quadratic in the security parameter (for the RLWE instantiation), giving a bootstrapped scheme with quasi-quadratic per-gate computation overall.

5.2.1 Decryption as a Circuit of Quasi-Linear Size and Logarithmic Depth

Recall that the decryption function is $m = [[\langle \mathbf{c}, \mathbf{s} \rangle]_q]_2$. Suppose that we are given the “bits” (elements in R_2) of \mathbf{s} as input, and we want to compute $[\langle \mathbf{c}, \mathbf{s} \rangle]_q$ using an arithmetic circuit that has Add and Mult gates over R_2 . (When we bootstrap, of course we are given the bits of \mathbf{s} in encrypted form.) Note that we will run the decryption function homomorphically on level-0 ciphertexts – i.e., when q is small, only polynomial in the security parameter. What is the complexity of this circuit? Most importantly for our purposes, what is its depth and size? The answer is that we can perform decryption with $\tilde{O}(\lambda)$ computation and $O(\log \lambda)$ depth. Thus, in the RLWE instantiation, we can evaluate the decryption function *homomorphically* using our new scheme with quasi-quadratic computation. (For the LWE instantiation, the bootstrapping computation is quasi-quartic.)

First, let us consider the LWE case, where \mathbf{c} and \mathbf{s} are n -dimensional integer vectors. Obviously, each product $\mathbf{c}[i] \cdot \mathbf{s}[i]$ can be written as the sum of at most $\log q$ “shifts” of $\mathbf{s}[i]$. These horizontal shifts of $\mathbf{s}[i]$ use at most $2 \log q$ columns. Thus, $\langle \mathbf{c}, \mathbf{s} \rangle$ can be written as the sum of $n \cdot \log q$ numbers, where each number has $2 \log q$ digits. As discussed in [8], we can use the three-for-two trick, which takes as input three numbers in binary (of arbitrary length) and outputs (using constant depth) two binary numbers with the same sum. Thus, with $O(\log(n \cdot \log q)) = O(\log n + \log \log q)$ depth and $O(n \log^2 q)$ computation, we obtain two numbers with the desired sum, each having $O(\log n + \log q)$ bits. We can sum the final two numbers with $O(\log \log n + \log \log q)$ depth and $O(\log n + \log q)$ computation. So far, we have used depth $O(\log n + \log \log q)$ and $O(n \log^2 q)$ computation to compute $\langle \mathbf{c}, \mathbf{s} \rangle$. Reducing this value modulo q is an operation akin to division, for which there are circuits of size $\text{polylog}(q)$ and depth $\log \log q$. Finally, reducing modulo 2 just involves dropping the most significant bits. Overall, since we are interested only in the case where $\log q = O(\log \lambda)$, we have that decryption requires $\tilde{O}(\lambda)$ computation and depth $O(\log \lambda)$.

For the RLWE case, we can use the R_2 plaintext space to emulate the simpler plaintext space \mathbb{Z}_2 . Using \mathbb{Z}_2 , the analysis is basically the same as above, except that we mention that the DFT is used to multiply elements in R .

In practice, it would be useful to tighten up this analysis by reducing the polylogarithmic factors in the computation and the constants in the depth. Most likely this could be done by evaluating decryption using symmetric polynomials [8, 9] or with a variant of the “grade-school addition” approach used in the Gentry-Halevi implementation [10].

5.2.2 Bootstrapping Lazily

Bootstrapping is rather expensive computationally. In particular, the cost of bootstrapping a ciphertext is greater than the cost of a homomorphic operation by approximately a factor of λ . This suggests the question: can we lower per-gate computation of a bootstrapped scheme by bootstrapping *lazily* – i.e., applying the refresh procedure only at a $1/L$ fraction of the circuit levels for some well-chosen L [11]? Here we show that the answer is yes. By bootstrapping lazily for $L = \theta(\log \lambda)$, we can lower the per-gate computation by a logarithmic factor.

Let us present this result somewhat abstractly. Suppose that the per-gate computation for a L -level no-bootstrapping FHE scheme is $f(\lambda, L) = \lambda^{a_1} \cdot L^{a_2}$. (We ignore logarithmic factors in f , since they will not affect the analysis, but one can imagine that they add a very small ϵ to the exponent.) Suppose that bootstrapping a ciphertext requires a c -depth circuit. Since we want to be capable of evaluation depth L *after* evaluating the c levels need to bootstrap a ciphertext, the bootstrapping procedure needs to begin with ciphertexts that can be used in a $(c + L)$ -depth circuit. Consequently, let us say that the computation needed to bootstrap a ciphertext is $g(\lambda, c + L)$ where $g(\lambda, x) = \lambda^{b_1} \cdot x^{b_2}$. The overall per-gate computation is approximately $f(\lambda, L) + g(\lambda, c + L)/L$, a quantity that we seek to minimize.

We have the following lemma.

Lemma 10. *Let $f(\lambda, L) = \lambda^{a_1} \cdot L^{a_2}$ and $g(\lambda, L) = \lambda^{b_1} \cdot L^{b_2}$ for constants $b_1 > a_1$ and $b_2 > a_2 \geq 1$. Let $h(\lambda, L) = f(\lambda, L) + g(\lambda, c + L)/L$ for $c = \theta(\log \lambda)$. Then, for fixed λ , $h(\lambda, L)$ has a minimum for $L \in [(c - 1)/(b_2 - 1), c/(b_2 - 1)]$ – i.e., at some $L = \theta(\log \lambda)$.*

Proof. Clearly $h(\lambda, L) = +\infty$ at $L = 0$, then it decreases toward a minimum, and finally it eventually increases again as L goes toward infinity. Thus, $h(\lambda, L)$ has a minimum at some positive value of L . Since $f(\lambda, L)$ is monotonically increasing (i.e., the derivative is positive), the minimum must occur where the derivative of $g(\lambda, c + L)/L$ is negative. We have

$$\begin{aligned} \frac{d}{dL} g(\lambda, c + L)/L &= g'(\lambda, c + L)/L - g(\lambda, c + L)/L^2 \\ &= b_2 \cdot \lambda^{b_1} \cdot (c + L)^{b_2-1}/L - \lambda^{b_1} \cdot (c + L)^{b_2}/L^2 \\ &= (\lambda^{b_1} \cdot (c + L)^{b_2-1}/L^2) \cdot (b_2 \cdot L - c - L), \end{aligned}$$

which becomes positive when $L \geq c/(b_2 - 1)$ – i.e., the derivative is negative only when $L = O(\log \lambda)$. For $L < (c - 1)/(b_2 - 1)$, we have that the above derivative is less than $-\lambda^{b_1} \cdot (c + L)^{b_2-1}/L^2$, which dominates the positive derivative of f . Therefore, for large enough value of λ , the value $h(\lambda, L)$ has its minimum at some $L \in [(c - 1)/(b_2 - 1), c/(b_2 - 1)]$. \square

This lemma basically says that, since homomorphic decryption takes $\theta(\log \lambda)$ levels and its cost is super-linear and dominates that of normal homomorphic operations (FHE.Add and FHE.Mult), it makes sense to bootstrap lazily – in particular, once every $\theta(\log \lambda)$ levels. (If one bootstrapped even more lazily than this, the super-linear cost of bootstrapping begins to ensure that the (amortized) per-gate cost of bootstrapping alone is increasing.) It is easy to see that, since the per-gate computation is dominated by bootstrapping, bootstrapping lazily every $\theta(\log \lambda)$ levels reduces the per-gate computation by a factor of $\theta(\log \lambda)$.

5.3 Batching the Bootstrapping Operation

Suppose that we are evaluating a circuit homomorphically, that we are currently at a level in the circuit that has at least d gates (where d is the dimension of our ring), and that we want to bootstrap (refresh) all of the ciphertexts corresponding to the respective wires at that level. That is, we want to homomorphically evaluate the decryption function at least d times in parallel. This seems like an ideal place to apply batching.

However, there are some nontrivial problems. In Section 5.1, our focus was rather limited. For example, we did not consider whether homomorphic operations could continue after the batched computation. Indeed, at first glance, it would appear that homomorphic operations *cannot* continue, since, after batching, the encrypted data is partitioned into non-interacting relatively-prime plaintext slots, whereas the whole point of homomorphic encryption is that the encrypted data can interact (within a common plaintext slot). Similarly, we did not consider homomorphic operations *before* the batched computation. Somehow, we need the input to the batched computation to come pre-partitioned into the different plaintext slots.

What we need are Pack and Unpack functions that allow the batching procedure to interface with “normal” homomorphic operations. One may think of the Pack and Unpack functions as an on-ramp to and an exit-ramp from the “fast lane” of batching. Let us say that normal homomorphic operations will always use the plaintext slot R_{p_1} . Roughly, the Pack function should take a bunch of ciphertexts c_1, \dots, c_d that encrypt messages $m_1, \dots, m_d \in \mathbb{Z}_p$ under key s_1 for modulus q and plaintext slot R_{p_1} , and then *aggregate* them into a single ciphertext c under some possibly different key s_2 for modulus q and plaintext slot $R_p = \bigotimes_{i=1}^d R_{p_i}$, so that correctness holds with respect to all of the different plaintext slots – i.e. $m_i = [[\langle c, s_2 \rangle]_q]_{p_i}$ for all i . The Pack function thus allows normal homomorphic operations to feed into the batch operation.

The Unpack function should accept the output of a batched computation, namely a ciphertext \mathbf{c}' such that $m_i = [[\langle \mathbf{c}', \mathbf{s}'_1 \rangle]_q]_{\mathbf{p}_i}$ for all i , and then de-aggregate this ciphertext by outputting ciphertexts $\mathbf{c}'_1, \dots, \mathbf{c}'_d$ under some possibly different common secret key \mathbf{s}'_2 such that $m_i = [[\langle \mathbf{c}'_i, \mathbf{s}'_2 \rangle]_q]_{\mathbf{p}_1}$ for all i . Now that all of the ciphertexts are under a common key and plaintext slot, normal homomorphic operations can resume. With such Pack and Unpack functions, we could indeed batch the bootstrapping operation. For circuits of large width (say, at least d) we could reduce the per-gate bootstrapping computation by a factor of d , making it only quasi-linear in λ . Assuming the Pack and Unpack functions have complexity at most quasi-quadratic in d (per-gate this is only quasi-linear, since Pack and Unpack operate on d gates), the overall per-gate computation of a batched-bootstrapped scheme becomes only quasi-linear.

Here, we describe suitable Pack and Unpack functions. These functions will make heavy use of the automorphisms $\sigma_{i \rightarrow j}$ over R that map elements of \mathbf{p}_i to elements of \mathbf{p}_j . (See Section 5.1.1.) We note that Smart and Vercauteren [21] used these automorphisms to construct something similar to our Pack function (though for unpacking they resorted to bootstrapping). We also note that Lyubashevsky, Peikert and Regev [14] used these automorphisms to permute the ideal factors \mathbf{q}_i of the modulus q , which was an essential tool toward their proof of the pseudorandomness of RLWE.

Toward Pack and Unpack procedures, our main idea is the following. If m is encoded in the free term as a number in $\{0, \dots, p-1\}$ and if $m = [[\langle \mathbf{c}, \mathbf{s} \rangle]_q]_{\mathbf{p}_i}$, then $m = [[\langle \sigma_{i \rightarrow j}(\mathbf{c}), \sigma_{i \rightarrow j}(\mathbf{s}) \rangle]_q]_{\mathbf{p}_j}$. That is, we can switch the plaintext slot but leave the decrypted message unchanged by applying the same automorphism to the ciphertext and the secret key. (These facts follow from the fact that $\sigma_{i \rightarrow j}$ is a homomorphism, that it maps elements of \mathbf{p}_i to elements of \mathbf{p}_j , and that it fixes free terms.) Of course, then we have a problem: the ciphertext is now under a different key, whereas we may want the ciphertext to be under the same key as other ciphertexts. To get the ciphertexts to be back under the same key, we simply use the SwitchKey algorithm to switch all of the ciphertexts to a new common key.

Some technical remarks before we describe Pack and Unpack more formally: We mention again that E.PublicKeyGen is modified in the obvious way so that $\mathbf{A} \cdot \mathbf{s} = p \cdot \mathbf{e}$ rather than $2 \cdot \mathbf{e}$, and that this modification induces a similar modification in SwitchKeyGen. Also, let $u \in R$ be a short element such that $u \in 1 + \mathbf{p}_1$ and $u \in \mathbf{p}_j$ for all $j \neq 1$. It is obvious that such a u with coefficients in $(-p/2, p/2]$ can be computed efficiently by first picking *any* element u' such that $u' \in 1 + \mathbf{p}_1$ and $u' \in \mathbf{p}_j$ for all $j \neq 1$, and then reducing the coefficients of u' modulo p .

PackSetup($\mathbf{s}_1, \mathbf{s}_2$): Takes as input two secret keys $\mathbf{s}_1, \mathbf{s}_2$. For all $i \in [1, d]$, it runs $\tau_{\sigma_{1 \rightarrow i}(\mathbf{s}_1) \rightarrow \mathbf{s}_2} \leftarrow \text{SwitchKeyGen}(\sigma_{1 \rightarrow i}(\mathbf{s}_1), \mathbf{s}_2)$.

Pack($\{\mathbf{c}_i\}_{i=1}^d, \{\tau_{\sigma_{1 \rightarrow i}(\mathbf{s}_1) \rightarrow \mathbf{s}_2}\}_{i=1}^d$): Takes as input ciphertexts $\mathbf{c}_1, \dots, \mathbf{c}_d$ such that $m_i = [[\langle \mathbf{c}_i, \mathbf{s}_1 \rangle]_q]_{\mathbf{p}_1}$ and $0 = [[\langle \mathbf{c}_i, \mathbf{s}_1 \rangle]_q]_{\mathbf{p}_j}$ for all $j \neq 1$, and also some auxiliary information output by PackSetup. For all i , it does the following:

- Computes $\mathbf{c}_i^* \leftarrow \sigma_{1 \rightarrow i}(\mathbf{c}_i)$. (Observe: $m_i = [[\langle \mathbf{c}_i^*, \sigma_{1 \rightarrow i}(\mathbf{s}_1) \rangle]_q]_{\mathbf{p}_i}$ while $0 = [[\langle \mathbf{c}_i^*, \sigma_{1 \rightarrow i}(\mathbf{s}_1) \rangle]_q]_{\mathbf{p}_j}$ for all $j \neq i$.)
- Runs $\mathbf{c}_i^\dagger \leftarrow \text{SwitchKey}(\tau_{\sigma_{1 \rightarrow i}(\mathbf{s}_1) \rightarrow \mathbf{s}_2}, \mathbf{c}_i^*)$ (Observe: Assuming the noise does not wrap, we have that $m_i = [[\langle \mathbf{c}_i^\dagger, \mathbf{s}_2 \rangle]_q]_{\mathbf{p}_i}$ and $0 = [[\langle \mathbf{c}_i^\dagger, \mathbf{s}_2 \rangle]_q]_{\mathbf{p}_j}$ for all $j \neq i$.)

Finally, it outputs $\mathbf{c} \leftarrow \sum_{i=1}^d \mathbf{c}_i^\dagger$. (Observe: Assuming the noise does not wrap, we have that $m_i = [[\langle \mathbf{c}, \mathbf{s}_2 \rangle]_q]_{\mathbf{p}_i}$ for all i .)

UnpackSetup($\mathbf{s}_1, \mathbf{s}_2$): Takes as input two secret keys $\mathbf{s}_1, \mathbf{s}_2$. For all $i \in [1, d]$, it runs $\tau_{\sigma_{i \rightarrow 1}(\mathbf{s}_1) \rightarrow \mathbf{s}_2} \leftarrow \text{SwitchKeyGen}(\sigma_{i \rightarrow 1}(\mathbf{s}_1), \mathbf{s}_2)$.

$\text{Unpack}(\mathbf{c}, \{\tau_{\sigma_{i \rightarrow 1}(\mathbf{s}_1) \rightarrow \mathbf{s}_2}\}_{i=1}^d)$: Takes as input a ciphertext \mathbf{c} such that $m_i = [[\langle \mathbf{c}, \mathbf{s}_1 \rangle]]_q]_{p_i}$ for all i , and also some auxiliary information output by UnpackSetup . For all i , it does the following:

- Computes $\mathbf{c}_i \leftarrow u \cdot \sigma_{i \rightarrow 1}(\mathbf{c})$. (Observe: Assuming the noise does not wrap, $m_i = [[\langle \mathbf{c}_i, \sigma_{i \rightarrow 1}(\mathbf{s}_1) \rangle]]_q]_{p_1}$ and $0 = [[\langle \mathbf{c}_i, \sigma_{i \rightarrow 1}(\mathbf{s}_1) \rangle]]_q]_{p_j}$ for all $j \neq 1$.)
- Outputs $\mathbf{c}_i^* \leftarrow \text{SwitchKey}(\tau_{\sigma_{i \rightarrow 1}(\mathbf{s}_1) \rightarrow \mathbf{s}_2}, \mathbf{c}_i)$. (Observe: Assuming the noise does not wrap, $m_i = [[\langle \mathbf{c}_i^*, \mathbf{s}_2 \rangle]]_q]_{p_1}$ and $0 = [[\langle \mathbf{c}_i^*, \mathbf{s}_2 \rangle]]_q]_{p_j}$ for all $j \neq 1$.)

Splicing the Pack and Unpack procedures into our scheme FHE is tedious but pretty straightforward. Although these procedures introduce many more encrypted secret keys, this does not cause a circular security problem as long as the chain of encrypted secret keys is acyclic; then the standard hybrid argument applies. After applying Pack or Unpack, one may apply modulus reduction to reduce the noise back down to normal.

5.4 More Fun with Funky Plaintext Spaces

In some cases, it might be nice to have a plaintext space isomorphic to \mathbb{Z}_p for some *large* prime p – e.g., one *exponential* in the security parameter. So far, we have been using R_p as our plaintext space, and (due to the rounding step in modulus switching) the size of the noise after modulus switching is proportional to p . When p is exponential, our previous approach for handling the noise (which keeps the magnitude of the noise polynomial in λ) obviously breaks down.

To get a plaintext space isomorphic to \mathbb{Z}_p that works for exponential p , we need a new approach. Instead of using an integer modulus, we will use an *ideal* modulus I (an ideal of R) whose *norm* is some large prime p , but such that we have a basis B_I of I that is very short – e.g. $\|B_I\| = O(\text{poly}(d) \cdot p^{1/d})$. Using an ideal plaintext space forces us to modify the modulus switching technique nontrivially.

Originally, when our plaintext space was R_2 , each of the moduli in our “ladder” was odd – that is, they were all congruent to each other modulo 2 and relatively prime to 2. Similarly, we will have to choose each of the moduli in our new ladder so that they are all congruent to each other modulo I . (This just seems necessary to get the scaling to work, as the reader will see shortly.) This presents a difficulty, since we wanted the norm of I to be large – e.g., exponential in the security parameter. If we choose our moduli q_j to be *integers*, then we have that the integer $q_{j+1} - q_j \in I$ – in particular, $q_{j+1} - q_j$ is a multiple of I ’s norm, implying that the q_j ’s are exponential in the security parameter. Having such large q_j ’s does not work well in our scheme, since the underlying lattice problems becomes easy when q_j/B is exponential in d where B is a bound of the noise distribution of fresh ciphertexts, and since we need B to remain quite small for our new noise management approach to work effectively. So, instead, our ladder of moduli will also consist of ideals – in particular, principle ideals (q_j) generated by an element of $q_j \in R$. Specifically, it is easy to generate a ladder of q_j ’s that are all congruent to 1 modulo I by sampling appropriately-sized elements q_j of the coset $1 + I$ (using our short basis of I), and testing whether the principal ideal (q_j) generated by the element has appropriate norm.

Now, let us reconsider modulus switching in light of the fact that our moduli are now principal ideals. We need an analogue of Lemma 4 that works for ideal moduli.

Let us build up some notation and concepts that we will need in our new lemma. Let \mathcal{P}_q be the half-open *parallelepiped* associated to the *rotation basis* of $q \in R$. The rotation basis \mathbf{B}_q of q is the d -dimensional basis formed by the coefficient vectors of the polynomials $x^i q(x) \bmod f(x)$ for $i \in [0, d-1]$. The associated parallelepiped is $\mathcal{P}_q = \{\sum z_i \cdot \mathbf{b}_i : \mathbf{b}_i \in \mathbf{B}_q, z_i \in [-1/2, 1/2)\}$. We need two concepts associated to this parallelepiped. First, we will still use the notation $[a]_q$, but where q is now an R -element rather than integer. This notation refers to a reduced modulo the *rotation basis* of a – i.e., the element $[a]_q$ such that $[a]_q - a \in qR$ and $[a]_q \in \mathcal{P}_q$. Next, we need notions of the *inner radius* $r_{q,in}$ and *outer radius* $r_{q,out}$ of \mathcal{P}_q – that is, the

largest radius of a ball that is circumscribed by \mathcal{P}_q , and the smallest radius of a ball that circumscribes \mathcal{P}_q . It is possible to choose q so that the ratio $r_{q,out}/r_{q,in}$ is $\text{poly}(d)$. For example, this is true when q is an integer. For a suitable value of $f(x)$ that determines our ring, such as $f(x) = x^d + 1$, the expected value of ratio will be $\text{poly}(d)$ even if q is sampled uniformly (e.g., according to discrete Gaussian distribution centered at 0). More generally, we will refer to $r_{\mathbf{B},out}$ as the outer radius associated to the parallelepiped determined by basis \mathbf{B} . Also, in the field $\mathbb{Q}(x)/f(x)$ overlying this ring, it will be true with overwhelming probability, if q is sampled uniformly, that $\|q^{-1}\| = 1/\|q\|$ up to a $\text{poly}(d)$ factor. For convenience, let $\alpha(d)$ be a polynomial such that $\|q^{-1}\| = 1/\|q\|$ up to a $\alpha(d)$ factor and moreover $r_{q,out}/r_{q,in}$ is at most $\alpha(d)$ with overwhelming probability. For such an α , we say q is α -good. Finally, in the lemma, γ_R denotes the expansion factor of R – i.e., $\max\{\|\mathbf{a} \cdot \mathbf{b}\|/\|\mathbf{a}\|\|\mathbf{b}\| : \mathbf{a}, \mathbf{b} \in R\}$.

Lemma 11. *Let q_1 and q_2 , $\|q_1\| < \|q_2\|$, be two α -good elements of R . Let \mathbf{B}_I be a short basis (with outer radius $r_{\mathbf{B}_I,out}$) of an ideal I of R such that $q_1 - q_2 \in I$. Let \mathbf{c} be an integer vector and $\mathbf{c}' \leftarrow \text{Scale}(\mathbf{c}, q_2, q_1, I)$ – that is, \mathbf{c}' is an R -element at most $2r_{\mathbf{B}_I,out}$ distant from $(q_1/q_2) \cdot \mathbf{c}$ such that $\mathbf{c}' - \mathbf{c} \in I$. Then, for any \mathbf{s} with*

$$\|[\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2}\| < \left(r_{q_2,in}/\alpha(d)^2 - (\|q_2\|/\|q_1\|)\gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \right) / (\alpha(d) \cdot \gamma_R^2)$$

we have

$$[\langle \mathbf{c}', \mathbf{s} \rangle]_{q_1} = [\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2} \bmod I \quad \text{and} \quad \|[\langle \mathbf{c}', \mathbf{s} \rangle]_{q_1}\| < \alpha(d) \cdot \gamma_R^2 \cdot (\|q_1\|/\|q_2\|) \cdot \|[\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2}\| + \gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s})$$

where $\ell_1^{(R)}(\mathbf{s})$ is defined as $\sum_i \|\mathbf{s}[i]\|$.

Proof. We have

$$[\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2} = \langle \mathbf{c}, \mathbf{s} \rangle - kq_2$$

for some $k \in R$. For the same k , let

$$e_{q_1} = \langle \mathbf{c}', \mathbf{s} \rangle - kq_1 \in R$$

Note that $e_{q_1} = [\langle \mathbf{c}', \mathbf{s} \rangle]_{q_1} \bmod q_1$. We claim that $\|e_{q_1}\|$ is so small that $e_{q_1} = [\langle \mathbf{c}', \mathbf{s} \rangle]_{q_1}$. We have:

$$\begin{aligned} \|e_{q_1}\| &= \| -kq_1 + \langle (q_1/q_2) \cdot \mathbf{c}, \mathbf{s} \rangle + \langle \mathbf{c}' - (q_1/q_2) \cdot \mathbf{c}, \mathbf{s} \rangle \| \\ &\leq \| -kq_1 + \langle (q_1/q_2) \cdot \mathbf{c}, \mathbf{s} \rangle \| + \| \langle \mathbf{c}' - (q_1/q_2) \cdot \mathbf{c}, \mathbf{s} \rangle \| \\ &\leq \gamma_R \cdot \|q_1/q_2\| \cdot \|[\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2}\| + \gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \\ &\leq \gamma_R^2 \cdot \|q_1\| \cdot \|q_2^{-1}\| \cdot \|[\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2}\| + \gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \\ &\leq \alpha(d) \cdot \gamma_R^2 \cdot (\|q_1\|/\|q_2\|) \cdot \|[\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2}\| + \gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \end{aligned}$$

By the final expression above, we see that the magnitude of e_{q_1} may actually be less than the magnitude of e_{q_2} if $\|q_1\|/\|q_2\|$ is small enough. Let us continue with the inequalities, substituting in the bound for $\|[\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2}\|$:

$$\begin{aligned} \|e_{q_1}\| &\leq \alpha(d) \cdot \gamma_R^2 \cdot (\|q_1\|/\|q_2\|) \cdot \left(r_{q_2,in}/\alpha(d)^2 - (\|q_2\|/\|q_1\|)\gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \right) / (\alpha(d) \cdot \gamma_R^2) \\ &\quad + \gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \\ &\leq (\|q_1\|/\|q_2\|) \cdot \left(r_{q_2,in}/\alpha(d)^2 - (\|q_2\|/\|q_1\|)\gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \right) + \gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \\ &\leq \left(r_{q_1,in} - \gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \right) + \gamma_R \cdot 2r_{\mathbf{B}_I,out} \cdot \ell_1^{(R)}(\mathbf{s}) \\ &= r_{q_1,in} \end{aligned}$$

Since $\|e_{q_1}\| < r_{q_1, in}$, e_{q_1} is inside the parallelepiped \mathcal{P}_{q_1} and it is indeed true that $e_{q_1} = [\langle \mathbf{c}', \mathbf{s} \rangle]_{q_1}$. Furthermore, we have $[\langle \mathbf{c}', \mathbf{s} \rangle]_{q_1} = e_{q_1} = \langle \mathbf{c}', \mathbf{s} \rangle - kq_1 = \langle \mathbf{c}, \mathbf{s} \rangle - kq_2 = [\langle \mathbf{c}, \mathbf{s} \rangle]_{q_2} \bmod I$. \square

The bottom line is that we can apply the modulus switching technique to moduli that are ideals, and this allows us to use, if desired, plaintext spaces that are very large (exponential in the security parameter) and that have properties that are often desirable (such as being isomorphic to a large prime field).

5.5 Other Optimizations

If one is willing to assume circular security, the keys $\{\mathbf{s}_j\}$ may all be the same, thereby permitting a public key of size independent of L .

While it is not necessary, squashing may still be a useful optimization in practice, as it can be used to lower the depth of the decryption function, thereby reducing the size of the largest modulus needed in the scheme, which may improve efficiency.

For the LWE-based scheme, one can use key switching to gradually reduce the dimension n_j of the ciphertext (and secret key) vectors as q_j decreases – that is, as one traverses to higher levels in the circuit. As q_j decreases, the associated LWE problem becomes (we believe) progressively harder (for a fixed noise distribution χ). This allows one to gradually reduce the dimension n_j without sacrificing security, and reduce ciphertext length faster (as one goes higher in the circuit) than one could simply by decreasing q_j alone.

6 Summary and Future Directions

Our RLWE-based FHE scheme without bootstrapping requires only $\tilde{O}(\lambda \cdot L^3)$ per-gate computation where L is the depth of the circuit being evaluated, while the bootstrapped version has only $\tilde{O}(\lambda^2)$ per-gate computation. For circuits of width $\Omega(\lambda)$, we can use batching to reduce the per-gate computation of the bootstrapped version by another factor of λ .

While these schemes should perform significantly better than previous FHE schemes, we caution that the polylogarithmic factors in the per-gate computation are large. One future direction toward a truly practical scheme is to tighten up these polylogarithmic factors considerably.

Acknowledgments. We thank Carlos Aguilar Melchor, Boaz Barak, Shai Halevi, Chris Peikert, Nigel Smart, and Jiang Zhang for helpful discussions and insights.

References

- [1] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.
- [2] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-DNF formulas on ciphertexts. In *Proceedings of Theory of Cryptography Conference 2005*, volume 3378 of *LNCS*, pages 325–342, 2005.
- [3] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. Manuscript, to appear in FOCS 2011, available at <http://eprint.iacr.org/2011/344>.
- [4] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. Manuscript, to appear in CRYPTO 2011.
- [5] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully-homomorphic encryption over the integers with shorter public-keys. Manuscript, to appear in Crypto 2011.

- [6] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *Advances in Cryptology - EUROCRYPT'10*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010. Full version available on-line from <http://eprint.iacr.org/2009/616>.
- [7] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.
- [8] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.
- [9] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. Manuscript, to appear in FOCS 2011, available at <http://eprint.iacr.org/2011/279>.
- [10] Craig Gentry and Shai Halevi. Implementing gentry’s fully-homomorphic encryption scheme. In *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 129–148. Springer, 2011.
- [11] Shai Halevi, 2011. Personal communication.
- [12] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594. Springer, 2007.
- [13] Kristin Lauter, Michael Naehrig, and Vinod Vaikuntanathan. Can homomorphic encryption be practical? Manuscript at <http://eprint.iacr.org/2011/405>, 2011.
- [14] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23, 2010.
- [15] Carlos Aguilar Melchor, Philippe Gaborit, and Javier Herranz. Additively homomorphic encryption with -operand multiplications. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 138–154. Springer, 2010.
- [16] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *STOC*, pages 333–342. ACM, 2009.
- [17] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.
- [18] Oded Regev. The learning with errors problem (invited survey). In *IEEE Conference on Computational Complexity*, pages 191–204. IEEE Computer Society, 2010.
- [19] Ron Rivest, Leonard Adleman, and Michael L. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–180, 1978.
- [20] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Public Key Cryptography - PKC'10*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.
- [21] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. Manuscript at <http://eprint.iacr.org/2011/133>, 2011.

- [22] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 377–394. Springer, 2010.

On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption

Adriana López-Alt
New York University

Eran Tromer
Tel Aviv University

Vinod Vaikuntanathan
MIT

Abstract

We propose a new notion of secure multiparty computation aided by a computationally-powerful but untrusted “cloud” server. In this notion that we call *on-the-fly multiparty computation (MPC)*, the cloud can *non-interactively* perform arbitrary, *dynamically chosen* computations on data belonging to arbitrary sets of users chosen *on-the-fly*. All user’s input data and intermediate results are protected from snooping by the cloud as well as other users. This extends the standard notion of fully homomorphic encryption (FHE), where users can only enlist the cloud’s help in evaluating functions on their own encrypted data.

In on-the-fly MPC, each user is involved only when initially uploading his (encrypted) data to the cloud, and in a final output decryption phase when outputs are revealed; the complexity of both is independent of the function being computed and the total number of users in the system. When users upload their data, they need not decide in advance which function will be computed, nor who they will compute with; they need only retroactively approve the eventually-chosen functions and on whose data the functions were evaluated.

This notion is qualitatively the best possible in minimizing interaction, since the users’ interaction in the decryption stage is inevitable: we show that removing it would imply generic program obfuscation and is thus impossible.

Our contributions are two-fold:

1. We show how on-the-fly MPC can be achieved using a new type of encryption scheme that we call *multikey FHE*, which is capable of operating on inputs encrypted under multiple, unrelated keys. A ciphertext resulting from a multikey evaluation can be jointly decrypted using the secret keys of all the users involved in the computation.
2. We construct a multikey FHE scheme based on NTRU, a very efficient public-key encryption scheme proposed in the 1990s. It was previously not known how to make NTRU fully homomorphic even for a single party. We view the construction of (multikey) FHE from NTRU encryption as a main contribution of independent interest. Although the transformation to a fully homomorphic system deteriorates the efficiency of NTRU somewhat, we believe that this system is a leading candidate for a practical FHE scheme.

Contents

1	Introduction	1
1.1	Our Results and Techniques	3
1.2	(Multikey) Fully Homomorphic Encryption from NTRU	4
1.3	On-The-Fly MPC from Multikey FHE	7
1.3.1	Protocol Security	7
1.4	Related Work	9
1.5	Roadmap	12
2	Definitions and Preliminaries	12
2.1	Notation	12
2.2	Σ -Protocols and Zero-Knowledge Proofs	12
2.3	Succinct Non-Interactive Arguments: SNARGs and SNARKs	15
2.3.1	Delegation of Computation from SNARGs	16
2.3.2	Constructions	16
2.4	Secure Multiparty Computation (MPC)	17
2.4.1	Security in the Ideal/Real Paradigm	17
2.4.2	Types of Adversaries	18
2.5	Fully Homomorphic Encryption (FHE)	19
2.5.1	Bootstrapping	21
2.6	Rings	21
2.6.1	Discrete Gaussians	22
2.6.2	The Ring LWE Assumption	23
2.6.3	Choice of Parameters	24
2.6.4	The Worst-case to Average-case Connection	24
2.7	NTRU Encryption	24
2.7.1	Security	25
3	Multikey FHE	26
3.1	Definition	26
3.2	Multikey FHE for a Small Number of Keys	27
3.2.1	$O(1)$ -Multikey FHE from any FHE	27
3.2.2	$O(\log \kappa)$ -Multikey FHE from Ring-LWE	30
3.3	Multikey Somewhat Homomorphic Encryption for Any Number of Keys	31
3.3.1	Multikey Homomorphism	31
3.3.2	Formal Description	33
3.3.3	Security	36
3.4	From Somewhat to Fully Homomorphic Encryption	36
3.4.1	Modulus Reduction	37
3.4.2	Obtaining A Leveled Homomorphic Scheme	38
3.4.3	Formal Description	39
3.4.4	Multikey Fully Homomorphic Encryption	42

4	On-the-Fly MPC from Multikey FHE	43
4.1	The Basic Protocol	43
4.1.1	Security Against Semi-Malicious Adversaries	44
4.2	Achieving Security Against Malicious Adversaries	46
4.2.1	Formal Protocol	49
4.2.2	Proof of Security	51
4.2.3	Efficient NIZKs to Prove Plaintext Knowledge	57
4.3	Impossibility of a 2-Round Protocol	60

1 Introduction

We are fast approaching a new digital era in which we store our data and perform our expensive computations remotely, on powerful servers — the “cloud”, in popular parlance. While the cloud offers numerous advantages in costs and functionality, it raises grave questions of confidentiality, since data stored in the cloud could be vulnerable to snooping by the cloud provider or even by other cloud clients [RTSS09]. Since this data often contains sensitive information (e.g., personal conversations, medical information and organizational secrets), it is prudent for the users to encrypt their data before storing it in the cloud. Recent advances in fully homomorphic encryption (FHE) [Gen09b, vDGHV10, BV11b, BV11a, GH11a, BGV12] make it possible to perform arbitrary computations on encrypted data, thus enabling the prospect of personal computers and mobile devices as trusted but weak interfaces to a powerful but untrusted cloud on which the bulk of computing is performed.

FHE is only suitable in settings where the computations involve a single user, since it requires inputs to be encrypted under the same key. However, there are many scenarios where users, who have uploaded their large data stores to the cloud in encrypted form, then decide to compute some joint function of their data. For example, they may wish the cloud to compute joint statistical information on their databases, locate common files in their collections, run a computational agent to reach a decision based on their pooled data (without leaking anything but the final decision), or generally, in contexts where multiple (mutually distrusting) users need to pool together their data to achieve a common goal.

The multiparty scenario is significantly more complex, and comes with a set of natural but stringent requirements. First, the participants involved in the computation and the function to be computed may be *dynamically chosen on-the-fly*, well after the data has been encrypted and uploaded to the cloud. Secondly, once the function is chosen, we should not expect the users to be online all the time, and consequently it is imperative that the cloud be able to perform the bulk of this computation (on the encrypted data belonging to the participants) *non-interactively*, without consulting the participants at all. Finally, all the burden of computation should indeed be carried by the cloud: the computational and communication complexity of the users should depend only on the size of the individual inputs and the output, and should be independent of both the complexity of the function computed and the total number of users in the system, both of which could be very large.

On-the-Fly Multiparty Computation. Consider a setting with a large universe of computationally weak users and a powerful cloud. An on-the-fly multiparty computation protocol proceeds thus:

1. The numerous users each encrypt their data and upload them to the cloud, *unaware of the identity or even the number* of other users in the system. Additional data may arrive directly to the cloud, encrypted under users’ public keys (e.g., as encrypted emails arriving to a cloud-based mailbox).
2. The cloud decides to evaluate an arbitrary *dynamically chosen* function on the data of arbitrary subset of users chosen *on-the-fly*. (The choice may be by some users’ request, or as a service to compute the function on the data of parties fulfilling some criterion, or by a need autonomously anticipated by the cloud provider, etc.) The cloud can perform this

computation *non-interactively*, without any further help from the users. The result is still encrypted.

3. The cloud and the subset of users whose data was used in the computation interact in a decryption phase. At this point the users retroactively approve the choice of function and the choice of peer users on whose data the function was evaluated, and cooperate to retrieve the output.

Crucially, the computation and communication of all the users (including the cloud) in the decryption phase should be *independent of* both the complexity of the function computed, and the size of the universe of parties (both of which can be enormous). Instead, the effort expended by the cloud and the users in this phase should depend only on the size of the output and the number of users who participated in the computation. Also crucially, the users need not be online at all during the bulk of the computation; they need to “wake up” only when it is time to decrypt the output.

We call this an *on-the-fly multiparty computation* (or *on-the-fly MPC* in short) to signify the fact that the functions to be computed on the encrypted data and the participants in the computation are both chosen on-the-fly and dynamically, without possibly even the knowledge of the participants. Protocols following this framework have additional desirable features such as the ability for users to “join” a computation asynchronously.

Possible Approaches (and Why They Do Not Work). The long line of work on secure multiparty computation (MPC) [GMW87, BGW88, CCD88, Yao82] does not seem to help us construct on-the-fly MPC protocols since the computational and communication complexities of *all the parties* in these protocols depends polynomially on the complexity of the function being computed.¹ In contrast, we are dealing with an asymmetric setting where the cloud computes a lot, but the users compute very little. (Nevertheless, we will use the traditional MPC protocols to interactively compute the decryption function at the end.)

Fully homomorphic encryption (FHE) is appropriate in such an asymmetric setting of computing with the cloud. Yet, traditional FHE schemes are *single-key* in the sense that they can perform (arbitrarily complex) computations on inputs encrypted under the same key. In our setting, since the parties do not trust each other, they will most certainly not want to encrypt their inputs using each other’s keys. Nevertheless, Gentry [Gen09a] proposed the following way of using single-key FHE schemes in order to do multiparty computation: first, the parties run a (short) MPC protocol to compute a joint public key, where the matching secret key is *secret-shared* among all the parties. The parties then encrypt their inputs under the joint public key and send the ciphertexts to the cloud who then uses the FHE scheme to compute an encryption of the result. Finally, the parties run yet another (short) MPC protocol to recover the result. A recent work by Asharov et al. [AJL⁺12] extends this schema and makes it efficient in terms of the concrete round, communication and computational complexity.

This line of work does not address the *dynamic* and *non-interactive* nature of on-the-fly MPC. In particular, once a subset of parties and a function are chosen, the protocols of [Gen09a, AJL⁺12] require the parties to be online and run an interactive MPC protocol to generate a joint public key. In contrast, we require that once the function and a subset of parties is chosen, the cloud performs

¹The works of Damgård et al. [DIK⁺08, DIK10] are an exception to this claim. However, it is not clear how to build upon these results to address the dynamic and non-interactive nature of on-the-fly MPC.

the (expensive) computations *non-interactively*, without help from any of the users. It would also be unsatisfactory to postpone the (lengthy) computation of the function until the interactive decryption phase; indeed, we require that once the users “wake up” for the decryption phase, the running time of all parties is independent of the complexity of the function being computed. Thus, even the feasibility of on-the-fly MPC is not addressed by existing techniques.

1.1 Our Results and Techniques

We present a *new notion* of fully homomorphic encryption (FHE) that we call a multikey FHE that permits computation on data encrypted under multiple unrelated keys; a *new construction* of multikey FHE based on the NTRU encryption scheme (originally proposed by Hoffstein, Pipher and Silverman [HPS98]); and a *new method* of achieving on-the-fly multiparty computation (for any a-priori bounded number of users) using a multikey FHE scheme. Although the number of users involved in any computation has to be bounded in our solution, the total number of users in the system is arbitrary.

Multikey FHE. An N -key fully homomorphic encryption scheme is the same as a regular FHE scheme with two changes. First, the homomorphic evaluation algorithm takes in polynomially many ciphertexts encrypted under at most N keys, together with the corresponding evaluation keys, and produces a ciphertext. Second, in order to decrypt the resulting ciphertext, one uses all the involved secret keys. As mentioned above, one of our main contributions is a construction of N -key FHE for any $N \in \mathbb{N}$ from the NTRU encryption scheme. We give an overview of our construction below (in Section 1.2) and refer the reader to Section 3.3 for more details.

Our NTRU-based construction raises a natural question: can any other FHE schemes be made multikey? We show that *any* FHE scheme is inherently a *multikey FHE* for a constant number of keys (in the security parameter), i.e. it can homomorphically evaluate functions on ciphertexts encrypted under at most a constant number of keys.² Furthermore, we show that the Ring-LWE based FHE scheme of Brakerski and Vaikuntanathan [BV11b] is multikey homomorphic for a logarithmic number of keys, but only for circuits of logarithmic depth. This arises from the fact that when multiple keys are introduced, it is no longer clear how to use relinearization or squashing to go beyond somewhat homomorphism. We refer the reader to Section 3.2 for more details.

On-the-Fly MPC from Multikey FHE. A multikey FHE scheme is indeed the right tool to perform on-the-fly MPC as shown by the following simple protocol: the users encrypt their inputs using their own public keys and send the ciphertexts to the cloud, the cloud then computes a dynamically chosen function on an arbitrary subset of parties using the multikey property of the FHE scheme, and finally, the users together run an interactive MPC protocol in order to decrypt. Note that the users can be offline during the bulk of the computation, and they need to participate only in the final cheap interactive decryption process. Note also that participants in the protocol need not be aware of the entire universe of users, but only those users that participate in a joint computation. This simple protocol provides us security against a semi-malicious collusion [AJW11, AJL⁺12] of the cloud with an arbitrary subset of parties. We then show how to achieve security

²This construction was originally suggested to us by an anonymous STOC 2012 reviewer; we include it here for completeness.

against a malicious adversary using zero-knowledge proofs and succinct argument systems [Kil92, Kil95, Mic94, GKR08, GLR11, BCCT12, BCCT13].

We further remark that the computation of the decryption function can itself be outsourced to the cloud. In particular, using the cloud-assisted MPC protocol of Asharov et al. [AJL⁺12] yields an on-the-fly MPC protocol with one offline round and 5 online rounds (for decryption).

We give an overview of our construction below (in Section 1.3) and refer the reader to Section 4 for more details.

Completely Non-Interactive On-the-Fly MPC? We know from the work of Halevi, Lindell, and Pinkas [HLP11] that in the non-interactive setting, the server can always evaluate the circuit multiple times, keeping some parties inputs but plugging in fake inputs of its choosing for the other parties. However, even if we accept this as the ideal functionality, we show that a non-interactive online phase *cannot* be achieved by drawing on the impossibility of general program obfuscation as a virtual black-box with single-bit output [BGI⁺01]. Thus, our notion is qualitatively “the best possible” in terms of interaction. Our techniques in showing this negative result are inspired by those of van Dijk and Juels [vDJ10]. We refer the reader to Section 4.3 for more details.

1.2 (Multikey) Fully Homomorphic Encryption from NTRU

The starting point for our main construction of multikey FHE is the NTRU encryption scheme of Hoffstein, Pipher, and Silverman [HPS98], with the modifications of Stehlé and Steinfeld [SS11b]. NTRU encryption is one of the earliest lattice-based cryptosystems, together with the Ajtai-Dwork cryptosystem [AD97] and the Goldreich-Goldwasser-Halevi cryptosystem [GGH97]. One of our most important contributions is to show that NTRU can be made fully homomorphic (for a single key)³ and moreover, that the resulting scheme can handle homomorphic evaluations on ciphertexts encrypted under any number of *different and independent* keys.

We find this contribution particularly interesting because NTRU was originally designed to be an *efficient* public-key encryption scheme, meant to replace RSA in applications where computational efficiency is at a premium (e.g. in applications that run on smart cards and embedded systems). Although the transformation to fully homomorphic encryption degrades the efficiency of the scheme, we believe it to be a *leading candidate* for a *practical* FHE scheme. Therefore, we view this as an important contribution of independent interest.

In this section we give an overview of our construction, and refer the reader to Section 3.3 for more details.

NTRU Encryption. We describe the modified NTRU scheme of Stehlé and Steinfeld [SS11b], which is based on the original NTRU cryptosystem [HPS98]. The scheme is parametrized by the ring $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/\langle x^n + 1 \rangle$, where n is a power of two, an odd prime number q , and a B -bounded distribution χ over R , for $B \ll q$. By “ B -bounded”, we mean that the magnitude of the coefficients of a polynomial sampled from χ is guaranteed to be less than B . We define $R_q \stackrel{\text{def}}{=} R/qR$, and use $[\cdot]_q$ to denote coefficient-wise reduction modulo q into the set $\{-\lfloor \frac{q}{2} \rfloor, \dots, \lfloor \frac{q}{2} \rfloor\}$.

³The observation that NTRU can be made *single-key* fully homomorphic was made concurrently by Gentry et al. [GHL⁺11].

- **Keygen**(1^κ): Key generation samples “small” polynomials $f', g \leftarrow \chi$ and sets $f \stackrel{\text{def}}{=} 2f' + 1$ so that $f \pmod{2} = 1$. If f is not invertible in R_q , it resamples f' . Otherwise, it computes the inverse f^{-1} of f in R_q and sets

$$\text{sk} = f \quad \text{and} \quad \text{pk} = [2gf^{-1}]_q$$

- **Enc**(pk, m): To encrypt a bit $m \in \{0, 1\}$, the encryption algorithm samples “small” polynomials $s, e \leftarrow \chi$, and outputs the ciphertext

$$c = [hs + 2e + m]_q$$

- **Dec**(sk, c): To decrypt a ciphertext c , the decryption algorithm computes $\mu = [fc]_q$ and returns $\mu \pmod{2}$.

Correctness follows from a few simple observations. First note that $[fc]_q = [2gs + 2fe + fm]_q$. Furthermore, since the elements g, s, f, e were all sampled from a B -bounded distribution and $B \ll q$, the magnitude of the coefficients in $2gs + 2fe + fm$ is smaller than $q/2$, so there is no reduction modulo q : in other words, $[2gs + 2fe + fm]_q = 2gs + 2fe + fm$. Therefore, $\mu = 2gs + 2fe + fm$. Taking modulo 2 yields the message m since by construction, $f \equiv 1 \pmod{2}$.

Multikey Homomorphism. We now briefly describe the (multikey) homomorphic properties of the scheme and the challenges encountered when converting it into a fully homomorphic encryption scheme.

Let $c_1 = [h_1s_1 + e_1 + m_1]_q$ and $c_2 = [h_2s_2 + e_2 + m_2]_q$ be ciphertexts under two different keys $h_1 = [2g_1f_1^{-1}]_q$ and $h_2 = [2g_2f_2^{-1}]_q$, respectively. We claim that $c_{\text{add}} \stackrel{\text{def}}{=} [c_1 + c_2]_q$ and $c_{\text{mult}} \stackrel{\text{def}}{=} [c_1c_2]_q$ decrypt to $m_1 + m_2$ and m_1m_2 respectively, under the *joint* secret key f_1f_2 . Indeed, notice that:

$$\begin{aligned} f_1f_2(c_1 + c_2) &= 2(f_1f_2e_1 + f_1f_2e_2 + f_2g_1s_1 + f_1g_2s_2) + f_1f_2(m_1 + m_2) \\ &= 2e_{\text{add}} + f_1f_2(m_1 + m_2) \end{aligned}$$

for a slightly larger noise element e_{add} . Similarly,

$$\begin{aligned} f_1f_2(c_1c_2) &= 2(2g_1g_2s_1s_2 + g_1s_1f_2(2e_2 + m_2) + g_2s_2f_1(2e_1 + m_1) + \\ &\quad f_1f_2(e_1m_2 + e_2m_1 + 2e_1e_2)) + f_1f_2(m_1m_2) \\ &= 2e_{\text{mult}} + f_1f_2(m_1m_2) \end{aligned}$$

for slightly larger noise element e_{mult} . This shows that the ciphertexts $c_{\text{add}} \stackrel{\text{def}}{=} [c_1 + c_2]_q$ and $c_{\text{mult}} \stackrel{\text{def}}{=} [c_1c_2]_q$ can be correctly decrypted to the sum and the product of the underlying messages, respectively, as long as the error does not grow too large.

Extending this to circuits, we notice that the secret key required to decrypt a ciphertext c that is the output of a homomorphic evaluation on ciphertexts encrypted under N different keys, is $\prod_{i=1}^N f_i^{d_i}$, where d_i is the degree of the i th variable in the polynomial function computed by the

circuit. Thus, decrypting a ciphertext that was the product of a homomorphic evaluation requires knowing the circuit! This is unacceptable even for somewhat homomorphic encryption.

We employ the relinearization technique of Brakerski and Vaikuntanathan [BV11a], to essentially reduce the degree from d_i to 1, so that the key needed to decrypt the evaluated ciphertext is now $\prod_{i=1}^N f_i$. This guarantees that decryption is dependent on the number of keys N but independent of the circuit computed. After using relinearization, we can show that the resulting scheme is multikey somewhat homomorphic for $\approx n^\delta$ keys and circuits of depth $\approx \log \log q - \delta \log n$ for any $\delta \in (0, 1)$.

From (Multikey) Somewhat to Fully Homomorphic Encryption. Once we obtain a (multikey) somewhat homomorphic encryption scheme, we can apply known techniques to convert it into a (multikey) fully homomorphic scheme. In particular, we follow the original template of our work [LTV12] and use modulus reduction [BV11a, BGV12] to increase the circuit depth that the scheme can handle in homomorphic evaluation. This yields a leveled homomorphic scheme for N keys that can evaluate circuits of depth D as long as $ND \approx \log q$. For any number of keys N and any depth D , we can set q to be large enough to guarantee the successful homomorphic evaluation of depth- D circuits on ciphertexts encrypted under N different keys.

Theorem 1.1 (Informal). *For all $N \in \mathbb{N}$ and $D \in \mathbb{N}$, there exists a leveled homomorphic encryption scheme that can homomorphically evaluate depth- D circuits on ciphertext encrypted under at most N different keys. The size of the keys and ciphertexts in the scheme grow polynomially with N and D .*

Finally, using an analog of Gentry’s bootstrapping theorem [Gen09b, Gen09a] for the multikey setting, we can convert the leveled homomorphic scheme into a fully homomorphic scheme, in which the algorithms are independent of the circuit depth D (albeit with an additional circular security assumption). On the other hand, we are unable to remove the dependence on the number of keys N , and therefore obtain a scheme that is fully homomorphic with respect to the depth of circuits it can evaluate, but “leveled” with respect to the number of different keys it can handle.

We remark that using the recent noise-management technique of Brakerski [Bra12], it is possible to obtain a simpler leveled homomorphic scheme, based on a weaker security assumption. This was already noted in the follow-up work of Bos et al. [BLLN13]. In another recent work, Gentry, Sahai, and Waters [GSW13] show how to remove the required evaluation key, yielding an even simpler scheme.

Security. Stehlé and Steinfeld [SS11b] showed that the security of the modified NTRU encryption scheme can be based on the Ring-LWE assumption of Lyubashevsky et al., which can be reduced to worst-case hard problems in ideal lattices [LPR10]. To prove the security of NTRU, Stehlé and Steinfeld first show that the public key $h = [2gf^{-1}]_q$ is *statistically* close to uniform over the ring R if f' and g are sampled from a discrete Gaussian with standard deviation $\text{poly}(n)\sqrt{q}$ (which can be shown to be a $\text{poly}(n)\sqrt{q}$ -bounded distribution). Unfortunately, if we sample f' and g from this distribution the error in a *single* homomorphic operation would grow large enough to cause decryption failures. We must therefore make the *assumption* that the public key $h = [2gf^{-1}]_q$ is

computationally indistinguishable⁴ from uniform over R when f' and g are sampled from a discrete Gaussian that is B -bounded for $B \ll q$.

Ultimately, we arrive at the following theorem.

Theorem 1.2 (Informal). *For all $N \in \mathbb{N}$, there exists a fully homomorphic encryption scheme that can perform homomorphic evaluation on ciphertext encrypted under at most N different keys. The size of the keys and ciphertexts in the scheme grow polynomially with N . The security of the scheme is based on the Ring-LWE assumption, the assumption that the public key is pseudorandom, and the assumption that the scheme is weakly circular secure.*

In a follow-up work, Bos et al. [BLLN13] show how to apply Brakerski's techniques [Bra12] to maintain the fully homomorphic properties of the scheme while sampling the elements f' and g from a discrete Gaussian with standard deviation $\text{poly}(n)\sqrt{q}$, as in the work of Stehlé and Steinfeld [SS11b]. This yields an NTRU-based FHE scheme that is secure under the RLWE assumption alone. However, as far as we know, this scheme is multikey for only a constant number of parties, which is an inherent property of any FHE scheme (see Section 3.2.1).

1.3 On-The-Fly MPC from Multikey FHE

Once we have constructed multikey FHE for any number of keys, we can construct on-the-fly MPC. The following gives an informal outline of our protocol.

Offline Phase: The clients sample independent key pairs $(\text{pk}_i, \text{sk}_i, \text{ek}_i)$, encrypt their input under their corresponding public key: $c_i \leftarrow \text{Enc}(\text{pk}_i, x_i)$, and send this ciphertext to the server along with the public and evaluation keys $(\text{pk}_i, \text{ek}_i)$.

Online Phase: Once a function has been chosen, together with a corresponding subset of computing parties V :

Step 1: The server performs the *multikey homomorphic evaluation* of the desired circuit on the corresponding ciphertexts, and broadcasts the evaluated ciphertext to all computing parties (i.e. all parties in V).

Step 2: The computing parties (i.e. parties in V) run a generic MPC protocol to decrypt the evaluated ciphertext using their individual secret keys sk_i .

Observe that the computation of the decryption function in Step 2 of the online phase can itself be delegated to the server. In particular, if we instantiate the decryption protocol using the cloud-assisted MPC protocol of Asharov et al. [AJW11, AJL⁺12] we obtain a round-efficient solution: the overall protocol has an online phase of only 5 rounds.

1.3.1 Protocol Security

We show that the above protocol is secure against *semi-malicious* adversaries [AJW11, AJL⁺12], who follow the protocol specifications (like semi-honest adversaries) but choose their random coins from an arbitrary distribution (like malicious adversaries). We then show how to modify the protocol to achieve security against malicious adversaries. We make three modifications, described below.

⁴It is not difficult to see that with our setting of parameters, the distribution of the public key is not statistically close to uniform. We must therefore rely on *computational* indistinguishability.

Modifying the Decryption Protocol. The first modification we make is to change the decryption protocol in Step 2 of the online phase to first check that the secret key being used is a *valid* secret key for the corresponding public and evaluation keys. This ensures that if decryption is successful, then in particular, a corrupted party knows a valid secret key $\tilde{\mathbf{sk}}_i$. This secret key *binds* the corrupted party to the input $\tilde{x}_i = \text{Dec}(\tilde{\mathbf{sk}}_i, \tilde{c}_i)$, which by semantic security of the FHE, must be independent of the honest inputs.

Once again, we note that the computation of this function can be delegated to the server using the cloud-assisted protocol of Asharov et al. [AJW11, AJL⁺12], yielding a 5-round online phase.

Adding Zero-Knowledge Proofs. We further require that in the offline phase, each party create a non-interactive zero-knowledge proof π_i^{ENC} showing that the ciphertext c_i is well-formed (i.e. that there exists plaintext x_i and randomness s_i such that $c_i = \text{Enc}(\mathbf{pk}_i, x_i; s_i)$). This guarantees that for a corrupted party, $\text{Dec}(\tilde{\mathbf{sk}}_i, \tilde{c}_i) \neq \perp$ and thus the party really “knows” an input \tilde{x}_i . Furthermore, it guarantees that the ciphertexts c_i are *fresh* encryptions, which is important in our setting of fully homomorphic encryption where we must ensure that the error stays low in a homomorphic evaluation.

While constructions of NIZK arguments are known for all of NP [GOS06, GOS12], using these constructions requires expensive NP reductions. To avoid this, in Section 4.2.3 we show how to construct an efficient NIZK argument system, secure in the random oracle model, for proving the well-formedness of a ciphertext in the NTRU-based multikey FHE scheme (the scheme we use to instantiate the generic multikey FHE scheme in our on-the-fly MPC construction).

Adding Verification of Computation. Finally, we must also rely on a succinct argument system [Kil92, Kil95, Mic94, GKR08, GLR11, BCCT12, BCCT13] to ensure that the server performs the homomorphic computation correctly. Due to the dynamic nature of our on-the-fly model, we are unable to use verifiable computation protocols in the pre-processing model [GGP10, CKV10, AIK10] or succinct arguments with a reference string that depends on the circuit being computed [Gro10, Lip12, GGPR13, PHGR13, Lip13]. These would require the clients to perform some pre-computation dependent on the circuit to be computed *before* knowing the circuit, or to interact with the server after a function has been selected and compute in time proportional to the circuit-size of the function. Indeed, the beauty of our on-the-fly MPC model is that the server can choose *any* function *dynamically*, “on-the-fly”, and homomorphically compute this function *without* interacting with the clients, who additionally, compute in time only polylogarithmically in the size of any function being computed.

We show how to guarantee verification of computation in two different cases.

Verification for Small Inputs: When the total size of the inputs (and therefore the ciphertexts) is small enough to be broadcasted to all parties, it suffices for the server to use any of the succinct arguments of [Kil92, Kil95, Mic94, GKR08, GLR11, BCCT12, BCCT13] to prove that it carried out the computation correctly as specified. Along with this argument, the server broadcasts the ciphertexts c_i and public and evaluations keys $(\mathbf{pk}_i, \mathbf{sk}_i)$ for all parties in V . With this information, the computing parties can verify the argument before engaging in the decryption protocol.

Verification for Large Inputs: In the case when the total size of the inputs (and therefore the ciphertexts) is too large to be broadcasted to all parties, then we additionally require the

parties to sample a hash key \mathbf{hk}_i for a collision-resistant hash function, and compute a digest d_i of the ciphertext c_i . Each party then sends the tuple $(\mathbf{pk}_i, \mathbf{ek}_i, c_i, \pi_i^{\text{ENC}}, \mathbf{hk}_i, d_i)$ to the server in the offline phase. It is then sufficient for the server to broadcast the tuples $(\mathbf{pk}_i, \mathbf{ek}_i, \mathbf{hk}_i, d_i)$ and a succinct argument for the NP language:

“there exist $\tilde{c}_1, \tilde{\pi}_1^{\text{ENC}}, \dots, \tilde{c}_N, \tilde{\pi}_N^{\text{ENC}}$ such that $d_i = H_{\mathbf{hk}_i}(\tilde{c}_i)$ and $c = \text{Eval}(C, (\tilde{c}_1, \mathbf{pk}_1, \mathbf{ek}_1), \dots, (\tilde{c}_N, \mathbf{pk}_N, \mathbf{ek}_N))$ and $\tilde{\pi}_i^{\text{ENC}}$ is a valid proof”.

If the succinct argument is additionally a *proof of knowledge*, as in the case of CS proofs [Mic94] under Valiant’s analysis [Val08], and the SNARKs of Bitansky et al. [BCCT12, BCCT13], then we are guaranteed that the server actually “knows” such $\tilde{c}_1, \tilde{\pi}_1^{\text{ENC}}, \dots, \tilde{c}_N, \tilde{\pi}_N^{\text{ENC}}$ whenever it successfully convinces the clients.

Putting everything together, we arrive at the following theorem.

Theorem 1.3 (Informal). *There exists an on-the-fly MPC protocol in the CRS-model with the following properties:*

- *Achieves security against malicious corruptions of an arbitrary subset of clients and possibly the server, under the Ring-LWE assumption, the assumption that the public key in the (modified) NTRU cryptosystem [HPS98, SS11b] is pseudorandom for a special setting of parameters, and the existence of zero-knowledge proofs and a secure succinct argument system.*
- *The offline phase runs in one (asynchronous) round of unidirectional communication from the parties to the server. The online phase runs in 5 rounds.*
- *The communication complexity of the online phase and the computation time of the computing parties therein is polylogarithmic in the size of the computation and the total size of the inputs, and linear in the size of their own input and the size of the output.*
- *The computation time of the server is polynomial in the size of the circuit.*

1.4 Related Work

We briefly survey related works in the areas of fully homomorphic encryption, MPC from homomorphic encryption, and MPC with the aid of a “cloud” server.

Fully Homomorphic Encryption. The notion of fully homomorphic encryption was first proposed by Rivest, Adleman, and Dertouzos [RAD78], but was only recently constructed in the groundbreaking result of Gentry [Gen09b, Gen09a]. In subsequent years, many improvements and new constructions have appeared in the literature [vDGHV10, BV11b, BV11a, BGV12, Bra12, BLLN13, GSW13, BV14].

Gentry’s first construction [Gen09b, Gen09a] followed the following blueprint: first, he constructed a *somewhat homomorphic* encryption scheme working over ideal lattices, that was able to perform a limited number of evaluations. He then proved a *bootstrapping theorem*, showing that if a somewhat homomorphic scheme can homomorphically evaluate its own decryption circuit, then it can be converted into a *fully homomorphic* scheme. Unfortunately, Gentry’s somewhat homomorphic scheme cannot evaluate its own decryption circuit and is therefore not bootstrappable. Nevertheless, he was able to construct a bootstrappable scheme by *squashing* the decryption

circuit sufficiently for the scheme to be able to homomorphically evaluate it. Using this squashing technique required making an additional security assumption, namely, the sparse subset sum (SSS) assumption.

van Dijk et al. [vDGHV10] subsequently showed how to construct FHE over the integers, and Brakerski and Vaikuntanathan [BV11b] showed how to construct FHE from the Ring-LWE assumption of Lyubashevsky, Regev, and Peikert [LPR10]. Both of these works use squashing and bootstrapping, as in Gentry’s original blueprint.

Gentry and Halevi [GH11a] showed how to use depth-3 arithmetic circuits and a hybrid of somewhat homomorphic encryption and multiplicatively homomorphic encryption (e.g. ElGamal encryption [Gam84]) to construct FHE without the use of squashing, and therefore without assuming the hardness of the SSS problem. In a separate work, Brakerski and Vaikuntanathan showed how to construct FHE from Regev’s (standard) LWE assumption [Reg05, Reg09]. In this work, they introduced the techniques of *relinearization* and *modulus reduction*, which allowed them to forgo squashing as well. Gentry, Brakerski, and Vaikuntanathan [BGV12] later showed a refinement of these techniques into so-called *key-switching* and *modulus switching*, and showed how to build “leveled” homomorphic schemes that can evaluate circuits of *any a-priori known depth* without the use of squashing or bootstrapping. Formally, they show that for every $D \in \mathbb{N}$, there exists a homomorphic scheme $\mathcal{E}^{(D)}$ that is able to homomorphically evaluate circuits of depth D . Their technique involves switching to a smaller modulus after every level in a homomorphic computation, therefore requiring a fairly large modulus at the start of the computation. This required basing security of their scheme on the hardness of solving approximate-SVP to within sub-exponential factors. Coron et al. [CNT12] show how to apply the modulus reduction technique over the integers.

In work subsequent to ours, Brakerski [Bra12] showed a new noise-management technique that forwent the modulus switching step, allowing the use of a single modulus that is much smaller than the one needed in the BGV scheme. Security of Brakerski’s scheme can be based on the hardness of solving approximate-SVP to within quasi-polynomial factors, a much weaker assumption. Bos et al. [BLLN13] show how to apply Brakerski’s noise-management technique to the (multikey) FHE described in this dissertation [LTV12], based on the NTRU encryption scheme of Hoffstein, Pipher, and Silverman [HPS98], with the modifications of Stehlé and Steinfeld [SS11b]. They further show that using these techniques, one can base security of the resulting FHE scheme on the Ring-LWE assumption alone, by using Stehlé and Steinfeld’s original analysis. Their construction, however, is multikey for only a constant number of keys, which we show is an inherent property of any FHE scheme. Coron et al. [CLT14] show how to apply Brakerski’s techniques over the integers.

Finally, Gentry et al. [GSW13] show how to construct a leveled homomorphic scheme that does not require the use of an evaluation key to perform homomorphic computation, as do all previous schemes. Brakerski and Vaikuntanathan [BV14] show how to leverage the techniques of Gentry et al. [GSW13] to build a leveled homomorphic scheme that is as secure as standard (non-FHE) LWE-based public-key encryption.

Many other works study the efficiency of the schemes described above and present several optimizations [SV10, SS11a, GH11b, CMNT11, GHPS12, GHS12a, GHS12b, GHS12c, CCK⁺13, SV14].

MPC from Homomorphic Encryption. The basic idea of using threshold homomorphic encryption (e.g. Paillier encryption [Pai99]) to boost the efficiency of MPC protocols was first presented by Cramer, Damgård, and Nielsen [CDN01], predating the existence of fully homomorphic

encryption (first showed by Gentry in 2009 [Gen09b, Gen09a]). They show that if the parties have access to a public key for an additively homomorphic encryption scheme, and if they also have a corresponding secret key secret-shared among them, then they can evaluate any Boolean circuit “under the covers” of the encryption. Using the homomorphic properties of the scheme, the parties can locally evaluate all addition gates. Cramer et al. additionally show a short, interactive subprotocol for evaluating multiplication gates. After showing the first construction of fully homomorphic encryption, Gentry used the same template to show a generic MPC construction from any FHE [Gen09a].

In a work concurrent to ours, Myers, Sergi, and Shelat [MSS13] show a *black-box* construction of MPC from any threshold FHE scheme. Their main hurdle is devising a way for parties to prove plaintext knowledge of a ciphertext. To this end, they present a 2-round protocol for proving plaintext knowledge, which they construct from any circuit-private FHE scheme. Their protocol is not zero-knowledge [GO94], but it conserves the semantic security of the ciphertext in question. They also show how to construct threshold FHE using the scheme of van Dijk et al. [vdGHV10] over the integers. While the communication of their protocol is independent of the circuit-size of the function being computed, their protocol is not computation-efficient: parties compute proportional to the complexity of the function.

Other works by Damgård et al. [BDOZ11, DPSZ12, DKL⁺13] build MPC from “semi-homomorphic” and somewhat homomorphic encryption. Their protocols require all parties to compute proportional to the complexity of the function at hand, and require interaction between parties at every gate. However, they display very good concrete efficiency. A work of Choudhury et al. [CLO⁺13] shows how to trade computation efficiency for communication efficiency. Their protocol is parametrized by an integer L . Setting $L = 2$ yields a classic MPC protocol, in which interaction is required for computing every gate. As L increases, interaction is required less frequently, and only to “refresh” the computation after an increasing number of steps. Thus, at their heart of their construction lies an interactive “bootstrapping” protocol that refreshes ciphertexts during the evaluation.

Finally, a recent work by Garg et al. [GGHR14] shows how to achieve 2-round MPC in the CRS model from indistinguishability obfuscation ($i\mathcal{O}$) [BGI⁺12]. As an optimization, they use multikey FHE (as defined in this work) to construct 2-round MPC with communication complexity that is independent of the circuit being computed. Though an efficient construction of $i\mathcal{O}$ is known for all circuits [GGH⁺13b], its security is based on assumptions on multilinear maps [GGH13a] that are not very well understood yet.

MPC on the Cloud. The idea of using a powerful cloud server to alleviate the computational efforts of parties in an MPC protocol was recently explored in the work on “server-aided MPC” by Kamara, Mohassel, and Raykova [KMR11]. Their protocols, however, require some of the parties to do a large amount of work, essentially proportional to the size of the computation.

Halevi, Lindell, and Pinkas [HLP11] recently considered the model of “secure computation on the web” wherein the goal is to minimize interaction between the parties. While their definition requires absolutely no interaction among the participants of the protocol (they only interact with the server), they show that this notion can only be achieved for a small class of functions. Our goal, on the other hand, is to construct MPC protocols for *arbitrary functions*.

1.5 Roadmap

We have given a high-level overview of our results. Detailed descriptions of all the results highlighted in this introduction can be found in the corresponding sections.

In Section 2 we present preliminaries, definitions and technical tools used throughout the remaining chapters.

In Section 3, we define multikey FHE and describe several constructions. In particular, we show that any FHE is inherently multikey for a constant number of keys, and that the ring-based FHE scheme of Brakerski and Vaikuntanathan is somewhat homomorphic for a logarithmic number of keys. More importantly, we show that the NTRU encryption scheme can be made multikey fully homomorphic for any number of keys.

In Section 4 we show how to construct on-the-fly MPC from multikey FHE. We show a basic protocol that is secure against semi-malicious corruptions, and then describe how to modify it to achieve security against malicious adversaries. We also show how to construct efficient NIZKs (in the random oracle model) for proving plaintext knowledge for the NTRU-based FHE scheme described in Section 3. Finally, we show that a completely non-interactive solution is impossible.

2 Definitions and Preliminaries

2.1 Notation

In this work, we use the following notation. We use κ to denote the security parameter. For an integer n , we use the notation $[n]$ to denote the set $[n] \stackrel{\text{def}}{=} \{1, \dots, n\}$. For a randomized function f , we write $f(x; r)$ to denote the unique output of f on input x with random coins r . We write $f(x)$ to denote a random variable for the output of $f(x; r)$ over uniformly random coins r . For a distribution or random variable X , we write $x \leftarrow X$ to denote the operation of sampling a random x according to X . For a set S , we overload notation and use $s \leftarrow S$ to denote sampling s from the uniform distribution over S . We use $y := f(x)$ to denote the deterministic evaluation of f on input x with output y . For two distributions, X and Y , we use $X \stackrel{c}{\approx} Y$ to mean that X and Y are computationally indistinguishable, and $X \stackrel{s}{\approx} Y$ to mean that they are statistically close.

2.2 Σ -Protocols and Zero-Knowledge Proofs

Σ -Protocols. We recall the notion of *gap* Σ -protocols [AJW11], a weaker version of Σ -protocols [CDS94], where honest-verifier zero-knowledge holds for all statements in some NP relation R_{zk} but soundness only holds w.r.t. $R_{\text{sound}} \supseteq R_{\text{zk}}$. In other words, zero-knowledge is guaranteed for an honest prover holding a statement in R_{zk} , but an honest verifier is only convinced that the statement is in a larger set $R_{\text{sound}} \supseteq R_{\text{zk}}$.

Definition 2.1 (Gap Σ -Protocol). *Let R_{zk} and R_{sound} be two NP relations such that $R_{\text{zk}} \subseteq R_{\text{sound}} \subseteq \{0, 1\}^* \times \{0, 1\}^*$, and let L_{zk} and L_{sound} be their corresponding NP languages. A gap Σ -protocol for $(R_{\text{zk}}, R_{\text{sound}})$ is a 3-step interactive protocol $\langle P, V \rangle$ between a prover $P = (P_1, P_2)$ and a verifier $V = (V_1, V_2)$, with the following syntax:*

- $(a, st) \leftarrow P_1(x, w)$: Given a statement and witness pair (x, w) , outputs a message a and a state string st .

- $c \leftarrow V_1(x, a)$: Given a statement x and message a , outputs a random challenge c from a challenge space \mathcal{C} .
- $z \leftarrow P_2(st, c)$: Given a state string st and a challenge c , outputs an answer z .
- $b \leftarrow V_2(x, a, c, z)$: Given a statement x , a message a , a challenge c , and an answer z , outputs a bit b , i.e. either accepts or rejects the transcript (a, c, z) for statement x .

We require that the following three properties hold:

Completeness: For any $(x, w) \in R_{\text{zk}}$,

$$\Pr \left[V_2(x, a, c, z) = 1 \mid \begin{array}{l} (a, st) \leftarrow P_1(x, w) \\ c \leftarrow V_1(x, a) \\ z \leftarrow P_2(st, c) \end{array} \right] = 1$$

Special Soundness: There exists an "extractor" such that for any two accepting transcripts (a, c, z) and (a, c', z') for the same statement x with $c \neq c'$, the extractor outputs a valid witness for $x \in R_{\text{sound}}$. Formally, there exists a PPT algorithm Ext such that for all x and all (a, c, z) and (a, c', z') such that $c \neq c'$ and $V_2(x, a, c, z) = V_2(x, a, c', z') = 1$:

$$\Pr [(x, w) \notin R_{\text{sound}} \mid w \leftarrow \text{Ext}(x, a, c, z, c', z')] = 1$$

Honest-Verifier Zero Knowledge (HVZK): There exists a PPT simulator Sim that "simulates" valid transcripts without knowing a witness, if it sees the challenge beforehand. Formally, there exists PPT algorithm Sim such that for all $(x, w) \in R_{\text{zk}}$ and all $c \in \mathcal{C}$, we have:

$$\left[(a, c, z) \mid \begin{array}{l} (a, st) \leftarrow P_1(x, w) \\ z \leftarrow P_2(st, c) \end{array} \right] \stackrel{s}{\approx} [(a', c, z') \mid (a', z') \leftarrow \text{Sim}(x, c)]$$

For an NP relation R with corresponding language L , a well-known construction using Σ -protocols allows a prover to show that either $x_0 \in L$ or $x_1 \in L$ without revealing which one holds. Suppose $\langle P, V \rangle$ is a Σ -protocol for R ; we construct a new protocol for proving that either $x_0 \in L$ or $x_1 \in L$. Let b be such that $(x_b, w_b) \in R$ for some witness w_b known to the prover. The prover chooses c_{1-b} at random from the challenge space \mathcal{C} and runs $(a_b, st) \leftarrow P_1(x_b, w_b)$ and $(a_{1-b}, z_{1-b}) \leftarrow \text{Sim}(x, c_{1-b})$. It sends (a_0, a_1) to the verifier, who returns a challenge c . The prover computes $c_b = c - c_{1-b}$, runs $z_b \leftarrow P_2(st, c)$ and sends (c_0, c_1, z_0, z_1) to the verifier, who checks that $V_2(x_0, a_0, c_0, z_0) = V_2(x_1, a_1, c_1, z_1) = 1$ and $c = c_0 + c_1$. The resulting protocol is called an *OR* Σ -protocol. The theorem below modifies this to the setting of *gap* Σ -protocols.

Theorem 2.1. Let R_{zk} and R_{sound} be two NP relations such that $R_{\text{zk}} \subseteq R_{\text{sound}} \subseteq \{0, 1\}^* \times \{0, 1\}^*$, and let $\langle P, V \rangle$ be a *gap* Σ -protocol for $(R_{\text{zk}}, R_{\text{sound}})$. The construction described above is a *gap OR* Σ -protocol for $(R_{\text{zk}}, R_{\text{sound}})$.

Non-Interactive Zero-Knowledge (NIZK). We also recall the notion of *non-interactive zero-knowledge (NIZK)* [BFM88]. For our purposes, it is more convenient to use the notion of (*same-string*) *NIZK arguments* from [SCO⁺01]. This definition and all our constructions that use it can be extended in the natural way to NIZK proofs, where soundness holds for all unbounded adversaries⁵.

Definition 2.2 (NIZK). *Let R be an NP relation on pairs (x, w) with corresponding language $L = \{x \mid \exists w \text{ s.t. } (x, w) \in R\}$. A non-interactive zero-knowledge (NIZK) argument system for R consists of three algorithms (Setup, Prove, Verify) with syntax:*

- $(\text{crs}, \text{tk}) \leftarrow \text{Setup}(1^\kappa)$: *Outputs a common reference string (CRS) crs and a trapdoor key tk to the CRS.*
- $\pi \leftarrow \text{Prove}_{\text{crs}}(x, w)$: *Outputs an argument π showing that $R(x, w) = 1$.*
- $0/1 \leftarrow \text{Verify}_{\text{crs}}(x, \pi)$: *Verifies whether or not the argument π is correct.*

For the sake of clarity, we write Prove and Verify without the crs in the subscript when the crs can be inferred from context. We require that the following three properties hold:

Completeness: *For any $(x, w) \in R$,*

$$\Pr \left[\text{Verify}(x, \pi) = 1 \mid \begin{array}{l} (\text{crs}, \text{tk}) \leftarrow \text{Setup}(1^\kappa) \\ \pi \leftarrow \text{Prove}(x, w) \end{array} \right] = 1$$

Soundness: *For any PPT adversary \tilde{P} ,*

$$\Pr \left[\begin{array}{l} \text{Verify}(x^*, \pi^*) = 1 \\ x^* \notin L \end{array} \mid \begin{array}{l} (\text{crs}, \text{tk}) \leftarrow \text{Setup}(1^\kappa) \\ (x^*, \pi^*) \leftarrow \tilde{P}(\text{crs}) \end{array} \right] = \text{negl}(\kappa).$$

Unbounded Zero-Knowledge: *There exists a PPT simulator Sim that “simulates” valid proofs without knowing a witness, but with the aid of the trapdoor key. We start by defining two oracles.*

The Prover Oracle: *A query to the prover oracle $\mathcal{P}(\cdot)$ consists of a pair (x, w) . The oracle checks if $(x, w) \in R$. If so, it outputs a valid argument $\text{Prove}(x, w)$; otherwise it outputs \perp .*

The Simulation Oracle: *A query to the simulation oracle $\text{SIM}_{\text{tk}}(\cdot)$ consists of a pair (x, w) . The oracle checks if $(x, w) \in R$. If so, it ignores w and outputs a simulated argument $\text{Sim}(\text{tk}, x)$; otherwise it outputs \perp .*

Formally, we require that for any PPT adversary \mathcal{A} , the advantage of \mathcal{A} in the following game is negligible (in κ):

- *The challenger samples $(\text{crs}, \text{tk}) \leftarrow \text{Setup}(1^\kappa)$ and gives crs to \mathcal{A} . The challenger also samples a bit $b \leftarrow \{0, 1\}$.*

⁵Apart from modifying the soundness condition, in the setting of proofs key generation samples a CRS but not a trapdoor, and the zero-knowledge simulator first samples a *simulated CRS* that is computationally indistinguishable from the real CRS, and a trapdoor to this CRS.

- If $b = 0$, the adversary \mathcal{A} is given access to the prover oracle $\mathcal{P}(\cdot)$. If $b = 1$, \mathcal{A} is given access to the simulation oracle $\mathcal{SIM}_{\text{tk}}(\cdot)$. In either case, the adversary can adaptively access its oracle..
- The adversary \mathcal{A} outputs a bit \tilde{b} .

The advtange of \mathcal{A} is defined to be $\left| \Pr[\tilde{b} = b] - \frac{1}{2} \right|$.

Fiat and Shamir [FS86] showed how to convert a Σ protocol $\langle P, V \rangle$ for an NP relation R into a NIZK argument for R secure in the random oracle model [BR93]. Informally, the CRS contains a description of a hash function H , which is modeled as a random oracle. To compute a non-interactive argument, the prover runs $(a, st) \leftarrow P_1(x, w)$ and obtains the verifier's challenge by applying the hash function to a and x : $c := H(a, x)$. It then computes $z \leftarrow P_2(st, c)$ and sends the argument $\pi = (a, c, z)$. The verifier runs $V_2(x, a, c, z)$ to verify the argument. The theorem below modifies this to the setting of *gap* Σ -protocols.

Theorem 2.2 ([FS86]). *Let R_{zk} and R_{sound} be two NP relations such that $R_{\text{zk}} \subseteq R_{\text{sound}} \subseteq \{0, 1\}^* \times \{0, 1\}^*$, and let $\langle P, V \rangle$ be a gap Σ -protocol for $(R_{\text{zk}}, R_{\text{sound}})$. Applying the Fiat-Shamir transform to $\langle P, V \rangle$ yields a non-interactive zero-knowledge (NIZK) argument system where soundness holds w.r.t. R_{sound} and completeness and zero-knowledge hold w.r.t. R_{zk} , secure in the random oracle model.*

Though secure in the random oracle model, we remark that in some cases standard-model security of the resulting NIZK appears to be harder to achieve [DJKL12, BDG⁺13]. In particular, if the language L is quasi-polynomially hard and the protocol has messages of size $\text{polylog}(\kappa)$ and is $\kappa^{\log \kappa}$ -HVZK, then the resulting NIZK cannot be proven sound via a black-box reduction to a (super-polynomially hard) falsifiable assumption [Nao03].

2.3 Succinct Non-Interactive Arguments: SNARGs and SNARKs

We review the definitions of succinct non-interactive arguments (SNARGs) and succinct non-interactive arguments of knowledge (SNARKs); we use the formalization of Gentry and Wichs [GW11], and Bitansky et al. [BCCT12]. As in the work of Bitansky et al., we allow the proof size to be polynomial in the size of the statement, but require it to be polylogarithmic in the size of the witness. We also require fast proof verification.

Definition 2.3 (SNARG). *Let R be an NP relation on pairs (x, w) with corresponding language $L = \{x \mid \exists w \text{ s.t. } (x, w) \in R\}$. A succinct non-interactive argument (SNARG) system for L consists of three algorithms (Setup, Prove, Verify) with syntax:*

- $(\text{vrs}, \text{priv}) \leftarrow \text{Setup}(1^\kappa)$: Outputs a verification reference string vrs and a private verification state priv .
- $\varphi \leftarrow \text{Prove}(\text{vrs}, x, w)$: Outputs an argument φ showing that $R(x, w) = 1$.
- $0/1 \leftarrow \text{Verify}(\text{priv}, x, \varphi)$: Verifies whether or not the argument φ is correct.

We require that the following properties hold:

Completeness: For any $(x, w) \in R$,

$$\Pr \left[\text{Verify}(\text{priv}, x, \varphi) = 1 \mid \begin{array}{l} (\text{vrs}, \text{priv}) \leftarrow \text{Setup}(1^\kappa) \\ \varphi \leftarrow \text{Prove}(\text{vrs}, x, w) \end{array} \right] = 1$$

In addition, $\text{Prove}(\text{vrs}, x, w)$ runs in time $\text{poly}(\kappa, |x|, |w|)$.

Adaptive Soundness: For any PPT adversary \tilde{P} ,

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{priv}, x^*, \varphi^*) = 1 \wedge \\ x^* \notin L \end{array} \mid \begin{array}{l} (\text{vrs}, \text{priv}) \leftarrow \text{Setup}(1^\kappa) \\ (x^*, \varphi^*) \leftarrow \tilde{P}(\text{vrs}) \end{array} \right] = \text{negl}(\kappa).$$

Succinctness: The length of the proof and the time required for its verification are polylogarithmic in the size of the witness, i.e. $\text{poly}(\kappa) (\text{poly}(|x|) + \text{polylog}(|w|))$.

Definition 2.4 (SNARK). A SNARG $\Phi = (\text{Setup}, \text{Prove}, \text{Verify})$ is additionally a proof of knowledge, or a succinct non-interactive argument of knowledge (SNARK) if it satisfies the following stronger definition of soundness:

Adaptive Extractability: There exists an extractor Ext that “extracts” a valid witness from any valid proof φ . Formally, for any PPT adversary \tilde{P} , there exists a PPT algorithm Ext such that:

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{priv}, x^*, \varphi^*) = 1 \wedge \\ R(x^*, w') = 0 \end{array} \mid \begin{array}{l} (\text{vrs}, \text{priv}) \leftarrow \text{Setup}(1^\kappa) \\ (x^*, \varphi^*) \leftarrow \tilde{P}(\text{vrs}) \\ w' \leftarrow \text{Ext}(x^*, \varphi^*) \end{array} \right] = \text{negl}(\kappa)$$

Public vs. Private Verifiability. In the case where $\text{priv} = \text{vrs}$, we say that the SNARG or SNARK is *publicly verifiable*. In this case, anyone can verify all proofs. Otherwise, we say that it is a *designated-verifier* SNARG/SNARK, in which case soundness/extractability is only guaranteed as long as priv remains secret to the prover. In this case, only the party holding priv can verify the proof.

2.3.1 Delegation of Computation from SNARGs

In delegation of computation we are concerned with at a client C , who wishes to delegate the computation of a pre-specified polynomial-time algorithm M on an input x , to a worker W . The client additionally wishes to verify the correctness of the output y returned by W (i.e. verify that $y = M(x)$) in time that is significantly smaller than the time required to compute $M(x)$ from scratch.

SNARGs can be used in this setting as follows: Define the NP language: $L_M = \{ (x, y) \text{ such that } M(x) = y \}$. A straight-forward witness to the statement $(x, y) \in L_M$ consists of the steps taken by M in a computation of $M(x)$ resulting in the output y . The size of this witness is proportional to the size of the computation. Using a SNARG guarantees that the size of the proof is polylogarithmic in the size of the witness, and therefore polylogarithmic in the size of the computation.

2.3.2 Constructions

Gentry and Wichs [GW11] proved that standard-model security of SNARGs with adaptive soundness and proof size sublinear in the witness and statement sizes, cannot be based on any falsifiable assumption [Nao03]. The constructions we show below either assume a random oracle [BR93] or most often use a non-falsifiable assumption.

CS Proofs. Kilian [Kil92, Kil95] showed how to perform succinct *interactive* verification for any NP language. His solution describes a 4-round protocol, where the prover first constructs a PCP for the correctness of the computation and then uses Merkle hashes to compress it to a sufficiently small proof. Micali’s CS proofs [Mic94] apply the Fiat-Shamir transform [FS86] to Kilian’s protocol, obtaining a non-interactive solution. CS proofs are publicly verifiable SNARGs (and SNARKs under Valiant’s analysis [Val08]); indeed, the only “setup” required is a description of a hash function H to use as the random oracle. This can be ensured by letting the \mathbf{vrs} be a random key for a (say) collision-resistant hash function.

Due to its use of the Fiat-Shamir transform, Micali’s solution is only secure in the *random oracle model* [BR93]. Unfortunately, several results have shown the implausibility of instantiating the random oracle in the Fiat-Shamir transform with any explicit hash function [HT98, Bar01, CGH04, DNRS03, GK03]. In particular, Dachman-Soled et al. [DJKL12, BDG⁺13] show that the security of CS proofs (even with non-adaptive soundness) cannot be based on any falsifiable assumption. On the other hand, it has been shown that the security of the Fiat-Shamir paradigm can be based on specific non-falsifiable assumptions regarding the existence of robust randomness condensers for seed-dependent sources [BLV06, DRV12].

Constructions with Small VRS. Bitansky et al. [BCCT12, BCCT13] and Goldwasser et al. [GLR11] revisit the construction of CS proofs and, based on the works of Di Crescenzo and Lipmaa [CL08] and Valiant [Val08], show how to construct SNARGs and SNARKs based on a different non-falsifiable assumption relating to the existence of extractable collision-resistant hash functions. In these works, the verifier’s entire computation (both in computing its reference string \mathbf{vrs} and in verifying the proof) depends only polylogarithmically in the size of the witness (i.e. the delegated computation). The SNARGs and SNARKs in these works are designated-verifier.

Allowing a Large VRS. Another series of works [Gro10, Lip12, GGPR13, PHGR13, Lip13] constructs SNARGs and SNARKs where the verifier’s reference string \mathbf{vrs} is allowed to depend on the circuit being delegated. In particular, Groth’s construction [Gro10] has a VRS of size quadratic in the circuit size. Lipmaa [Lip12] reduces this size to be quasi-linear, and the works of Gennaro et al. [GGPR13] and Parno et al. [PHGR13] further reduce it to linear in the circuit size. Lipmaa [Lip13] refines the construction of Gennaro et al. to reduce the magnitude of the constant in the size of the VRS. All of these constructions are based on certain number-theoretic non-falsifiable assumptions.

2.4 Secure Multiparty Computation (MPC)

Let f be an N -input function with single output. A *multiparty protocol* Π for f is a protocol between N interactive Turing Machines P_1, \dots, P_N , called *parties*, such that for all $\vec{x} = (x_1, \dots, x_N)$, the output of Π in an execution where P_i is given x_i as input, is $y \stackrel{\text{def}}{=} f(\vec{x})$.

2.4.1 Security in the Ideal/Real Paradigm

Informally, a multiparty protocol Π is *secure* if after running Π , no colluding set of corrupt parties can learn anything about an honest player’s input or change the output of an honest party. We formalize this in the Ideal/Real paradigm (see e.g. [Gol04]).

Ideal and Real Worlds. We define an *ideal world* in which the computation of f is performed through a trusted functionality \mathcal{F} that receives inputs x_i from each party P_i , computes $y \stackrel{\text{def}}{=} f(x_1, \dots, x_N)$ and gives y to all parties P_1, \dots, P_N . It is clear that in the ideal world, the only information that any party learns is its own input and the output y . We also define a *real world* in which parties P_1, \dots, P_N run the protocol Π .

The Network. We assume that the real-world execution of the protocol is performed over a *secure* and *synchronous* network; that is, we assume that parties can reliably send messages to other parties without these being read or altered in transmission, and that all point-to-point communications happen at the same time. We also assume that a secure *broadcast* channel is available to all parties.

The Adversary. In either world, we consider a single adversary that is allowed to corrupt any subset of $t < N$ parties. An adversary is modeled as an interactive Turing Machine that receives all messages directed to the corrupted parties and controls the messages sent by them. In this work, we consider only *static* adversaries, that is, adversaries that select the subset of corrupted parties non-adaptively, before any computation is performed. On the other hand, we assume that in each round of the protocol, the adversary chooses the messages for the corrupted parties adaptively, based on the entire transcript of the protocol, up to that round.

We remark that our results can be extended to achieve security against *rushing* real-world adversaries who, on any given round, choose the messages for the corrupted parties adaptively, based on the entire transcript of the protocol *and the messages of the honest parties on that round*. Note that rushing adversaries correspond to a semi-synchronous model of communication.

Output Distributions. We use $\text{IDEAL}_{\mathcal{F}, \mathcal{S}}(\vec{x})$ to denote the joint output of an ideal-world adversary \mathcal{S} and parties P_1, \dots, P_N in an ideal execution with functionality \mathcal{F} and inputs $\vec{x} = (x_1, \dots, x_N)$. Similarly, we use $\text{REAL}_{\Pi, \mathcal{A}}(\vec{x})$ to denote the joint output of a real-world adversary \mathcal{A} and parties P_1, \dots, P_N in an execution of protocol Π with inputs $\vec{x} = (x_1, \dots, x_N)$.

We say that a protocol Π *securely realizes* \mathcal{F} *against the class of adversaries* Adv , if for every real-world adversary $\mathcal{A} \in \text{Adv}$, there exists an ideal-world adversary \mathcal{S} with black-box access to \mathcal{A} such that for all input vectors \vec{x} ,

$$\text{IDEAL}_{\mathcal{F}, \mathcal{S}}(\vec{x}) \stackrel{c}{\approx} \text{REAL}_{\Pi, \mathcal{A}}(\vec{x})$$

2.4.2 Types of Adversaries

As stated above, in this work we only consider classes of adversaries Adv containing static adversaries that corrupt any subset of $t < N$ parties. We now describe three different types of adversaries: malicious, semi-honest, and semi-malicious. The first two are used extensively in the literature, while the latter was introduced recently by Asharov et al. [AJW11, AJL⁺12]. Of these, malicious adversaries are the strongest, and it is our end goal to achieve security against them in all our protocols.

It is customary to prove security against semi-honest adversaries as a stepping stone to proving security against malicious adversaries. However, in this work we follow a different path and first prove security against semi-malicious adversaries. We then show how to modify the protocol at hand to achieve security against malicious adversaries. For completeness, we describe all three

types of adversaries below and describe how security against one type is related to security against another.

Semi-Honest Adversaries. A semi-honest adversary, also known as an honest-but-curious adversary, is one that follows the protocol as described (samples randomness from the correct distribution, and computes the specified message at each round), but given its view of the protocol will try to learn information about honest players' inputs.

Malicious Adversaries. A malicious adversary is not restricted in how it samples random elements or how it computes its messages at each round. It can sample random elements from any arbitrary distribution, and compute the messages of corrupted parties in any arbitrary way, adaptively, according to the partial view it has seen up to that point.

Semi-Malicious Adversaries. Recall that an adversary is modeled as an interactive Turing Machine (ITM). A semi-malicious adversary is an ITM with an additional *witness tape*. At each round ℓ and for every corrupted party P_j , the adversary must write on the special witness tape, *some* witness pair $(x_j^{(\ell)}, r_j^{(\ell)})$ of input and randomness that explains the message $m_j^{(\ell)}$ sent by P_j on that round. More formally, the messages of a corrupted party P_j must match those of the specified honest protocol when at each round ℓ party P_j is run with input and randomness $(x_j^{(\ell)}, r_j^{(\ell)})$.

A semi-malicious adversary can sample random elements from any arbitrary distribution, but it must follow the correct behavior of the honest protocol with inputs and randomness that it *knows*. It is therefore weaker than a malicious adversary, who might not know witnesses for the messages it sends at every round, but stronger than a semi-honest adversary, whose witnesses at every round are distributed honestly.

From Semi-Malicious to Malicious Security. Asharov et al. [AJW11, AJL⁺12] show how to generically transform a protocol that is secure against semi-malicious adversaries into one that is secure against malicious adversaries. The idea behind the compiler is to have each party prove in zero-knowledge that every message it sends follows the honest protocol and is consistent with all previous messages. In particular, this forces all parties to know witnesses that explain their behavior at every round. The same compiler works in our security model with one subtlety: instead of using standard zero-knowledge proofs, the protocol must use *zero-knowledge proofs of knowledge*. This is to ensure that the simulator can extract the witness $w_j^{(\ell)}$ from the proof sent on round ℓ by the malicious adversary on behalf of the corrupted party P_j . We refer the reader to the work of Asharov et al. [AJW11, AJL⁺12] for more details.

Finally, we note that unlike the standard GMW compiler from semi-honest security to malicious security [GMW87], the parties are not required to perform any coin-flipping. This, in particular, reduces the round complexity of the resulting protocol.

2.5 Fully Homomorphic Encryption (FHE)

We review the definitions of fully and leveled homomorphic encryption.

Definition 2.5 (\mathcal{C} -Homomorphic Encryption [Gen09b]). For a class of circuits \mathcal{C} , a \mathcal{C} -homomorphic encryption scheme is a tuple of algorithms $\mathcal{E} = (\text{Setup}, \text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})$ with the following syntax:

- $\text{params} \leftarrow \text{Setup}(1^\kappa)$: For security parameter κ , outputs public parameters params . All other algorithms, $\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval}$, implicitly take params as input, even when not explicitly stated.
- $(\text{pk}, \text{sk}, \text{ek}) \leftarrow \text{Keygen}(1^\kappa)$: For a security parameter κ , outputs a public key pk , a secret key sk , and a (public) evaluation key ek .
- $c \leftarrow \text{Enc}(\text{pk}, m)$: Given a public key pk and a message m , outputs a ciphertext c .
- $m := \text{Dec}(\text{sk}, c)$: Given a secret key sk and a ciphertext c , outputs a message m .
- $c := \text{Eval}(\text{ek}, C, c_1, \dots, c_\ell)$: Given an evaluation key ek , a (description of a) circuit C and ℓ ciphertexts c_1, \dots, c_ℓ , outputs a ciphertext c .

We require that for all $c \in \mathcal{C}$, all $(\text{pk}, \text{sk}, \text{ek})$ in the support of $\text{Keygen}(1^\kappa)$ and all plaintexts (m_1, \dots, m_ℓ) and ciphertexts (c_1, \dots, c_ℓ) such that c_i is in the support of $\text{Enc}(\text{pk}, m_i)$, if $c := \text{Eval}(\text{ek}, C, c_1, \dots, c_\ell)$, then $\text{Dec}(\text{sk}, c) = C(m_1, \dots, m_\ell)$.

Definition 2.6 (Fully Homomorphic Encryption [Gen09b]). An encryption scheme \mathcal{E} is fully homomorphic if it satisfies the following properties:

Correctness: \mathcal{E} is \mathcal{C} -homomorphic for the class \mathcal{C} of all circuits.

Compactness: The computational complexity of \mathcal{E} 's algorithms is polynomial in the security parameter κ , and in the case of the evaluation algorithm, the size of the circuit.

We now state the definition of leveled homomorphic encryption from [BGV12], which is a relaxation of the original definition of fully homomorphic encryption (Definition 2.6). The main difference is that Definition 2.6 requires all algorithms (decryption in particular) to be independent of the circuit(s) that the scheme can evaluate. Leveled homomorphic encryption relaxes this definition to let all algorithms (including decryption) depend on the circuit depth D .

Definition 2.7 (Leveled Homomorphic Encryption [BGV12]). Let $\mathcal{C}^{(D)}$ be the class of all circuits of depth at most D (that use some specified complete set of gates). We say that a family of homomorphic encryption schemes $\{\mathcal{E}^{(D)} : D \in \mathbb{Z}^+\}$ is leveled fully homomorphic if, for all $D \in \mathbb{Z}^+$, it satisfies the following properties:

Correctness: $\mathcal{E}^{(D)}$ is $\mathcal{C}^{(D)}$ -homomorphic.

Compactness: The computational complexity of $\mathcal{E}^{(D)}$'s algorithms is polynomial in the security parameter κ and D , and in the case of the evaluation algorithm, the size of the circuit. We emphasize that this polynomial must be the same for all D .

2.5.1 Bootstrapping

We remind the reader of the definition of a bootstrappable encryption scheme and present Gentry's bootstrapping theorem [Gen09b, Gen09a] that states that a bootstrappable scheme can be converted into a fully homomorphic one.

Definition 2.8 (Bootstrappable Scheme). *Let $\mathcal{E} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})$ be a \mathcal{C} -homomorphic encryption scheme, and let f_{add} and f_{mult} be the augmented decryption functions of the scheme defined as*

$$f_{\text{add}}^{c_1, c_2}(\text{sk}_1, \dots, \text{sk}_N) = \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c_1) \quad \text{XOR} \quad \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c_2)$$

$$f_{\text{mult}}^{c_1, c_2}(\text{sk}_1, \dots, \text{sk}_N) = \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c_1) \quad \text{AND} \quad \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c_2)$$

\mathcal{E} is bootstrappable if $\{f_{\text{add}}^{c_1, c_2}, f_{\text{mult}}^{c_1, c_2}\}_{c_1, c_2} \subseteq \mathcal{C}$, namely, if it can homomorphically evaluate f_{add} and f_{mult} .

Definition 2.9 (Weak Circular Security). *A public-key encryption scheme $\mathcal{E} = (\text{Keygen}, \text{Enc}, \text{Dec})$ is weakly circular secure if it is IND-CPA secure even for an adversary with auxiliary information containing encryptions of all secret key bits: $\{\text{Enc}(\text{pk}, \text{sk}[i])\}_i$. Namely, no polynomial-time adversary can distinguish an encryption of 0 from an encryption of 1, even given this additional information.*

Theorem 2.3 (Bootstrapping Theorem). *Let \mathcal{E} be a bootstrappable scheme that is also weakly circular secure. Then there exists a fully homomorphic encryption scheme \mathcal{E}' .*

2.6 Rings

In this section we introduce preliminaries to our concrete constructions, which are all ring-based. Some of the discussion in this section is taken verbatim from the work of Brakerski and Vaikuntanathan [BV11b].

We work over rings $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$ and $R_q \stackrel{\text{def}}{=} R/qR$ for some degree $n = n(\kappa)$ integer polynomial $\phi(x) \in \mathbb{Z}[x]$ and a prime integer $q = q(\kappa) \in \mathbb{Z}$. Note that R_q is isomorphic to $\mathbb{Z}_q[x]/\langle \phi(x) \rangle$, the ring of degree n polynomials modulo $\phi(x)$ with coefficients in \mathbb{Z}_q . Addition in these rings is done component-wise in their coefficients (thus, their additive group is isomorphic to \mathbb{Z}^n and \mathbb{Z}_q^n respectively), and multiplication is polynomial multiplication modulo $\phi(x)$ (and also q , in the case of the ring R_q). An element in R (or R_q) can be viewed as a degree $(n-1)$ polynomial over \mathbb{Z} (or \mathbb{Z}_q). We represent such an element using the vector of its n coefficients. In the case of R_q each coefficient is in the range $\{-\lfloor \frac{q}{2} \rfloor, \dots, \lfloor \frac{q}{2} \rfloor\}$. For an element $a(x) = a_0 + a_1x + \dots + a_{n-1}x^{n-1} \in R$, we let $\|a\|_\infty = \max |a_i|$ denote its ℓ_∞ norm.

In this work, we set $\phi(x) = x^n + 1$ where n is a power of two, and use distributions over the ring $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$. For the purpose of homomorphism, the only important property of these distributions is the magnitude of the coefficients of a polynomial output by the distribution. Hence, we define a B -bounded distribution to be a distribution over R where the ℓ_∞ -norm of a sample is bounded by B .

Definition 2.10. (B -BOUNDED POLYNOMIAL) *A polynomial $e \in R$ is called B -bounded if $\|e\|_\infty \leq B$.*

Definition 2.11. (*B*-BOUNDED DISTRIBUTION) A distribution ensemble $\{\chi_\kappa\}_{\kappa \in \mathbb{N}}$, supported over R , is called *B*-bounded (for $B = B(\kappa)$) if for all e in the support of χ_κ , we have $\|e\|_\infty < B$. In other words, a *B*-bounded distribution over R outputs a *B*-bounded polynomial.

The following lemma says that multiplication in the ring $\mathbb{Z}[x]/\langle x^n + 1 \rangle$ increases the norm of the constituent elements only by a small amount.

Lemma 2.4. Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and let $R = \mathbb{Z}[x]/\langle \phi(x) \rangle$. For any $s, t \in R$,

$$\|s \cdot t\| \leq \sqrt{n} \cdot \|s\| \cdot \|t\| \quad \text{and} \quad \|s \cdot t\|_\infty \leq n \cdot \|s\|_\infty \cdot \|t\|_\infty$$

Lemma 2.4 yields the following corollary.

Corollary 2.5. Let $n \in \mathbb{N}$, let $\phi(x) = x^n + 1$ and $R = \mathbb{Z}[x]/\langle \phi(x) \rangle$. Let χ be a *B*-bounded distribution over the ring R and let $s_1, \dots, s_k \leftarrow \chi$. Then $s \stackrel{\text{def}}{=} \prod_{i=1}^k s_i$ is $(n^{k-1}B^k)$ -bounded.

2.6.1 Discrete Gaussians

For any real $r > 0$ the Gaussian function on \mathbb{R}^n centered at \mathbf{c} with parameter r is defined as:

$$\forall \mathbf{x} \in \mathbb{R}^n : \quad \rho_{r,\mathbf{c}}(\mathbf{x}) \stackrel{\text{def}}{=} e^{-\pi \|\mathbf{x} - \mathbf{c}\|^2 / r^2}$$

Definition 2.12. For any $n \in \mathbb{N}$ and for any $\mathbf{c} \in \mathbb{R}^n$ and real $r > 0$, the Discrete Gaussian distribution over \mathbb{Z}^n with standard deviation r and centered at \mathbf{c} is defined as:

$$\forall \mathbf{x} \in \mathbb{Z}^n : \quad D_{\mathbb{Z}^n, r, \mathbf{c}} \stackrel{\text{def}}{=} \frac{\rho_{r,\mathbf{c}}(\mathbf{x})}{\rho_{r,\mathbf{c}}(\mathbb{Z}^n)}$$

where $\rho_{r,\mathbf{c}}(\mathbb{Z}^n) \stackrel{\text{def}}{=} \sum_{\mathbf{x} \in \mathbb{Z}^n} \rho_{r,\mathbf{c}}(\mathbf{x})$ is a normalization factor.

We present some elementary facts about the Gaussian distribution. The first fact shows that the discrete Gaussian distribution over \mathbb{Z}^n with standard deviation r outputs a $(r\sqrt{n})$ -bounded polynomial with high probability. This allows us to define a *truncated* Gaussian distribution that is $(r\sqrt{n})$ -bounded and statistically close to the discrete Gaussian.

Lemma 2.6 (MR07). For any real number $r > \omega(\sqrt{\log n})$, we have

$$\Pr_{x \leftarrow D_{\mathbb{Z}^n, r}} [\|x\| > r\sqrt{n}] \leq 2^{-n+1}$$

Using Lemma 2.6 together with the fact that for all $\mathbf{x} \in \mathbb{R}^n$, $\|\mathbf{x}\| \geq \|\mathbf{x}\|_\infty$ yields the following bound.

Lemma 2.7. Let $n = \omega(\log \kappa)$. For any real number $r > \omega(\sqrt{\log n})$, we have

$$\Pr_{\mathbf{x} \leftarrow D_{\mathbb{Z}^n, r}} [\|\mathbf{x}\|_\infty > r\sqrt{n}] \leq 2^{-n+1} = \text{negl}(\kappa)$$

Define the *truncated Discrete Gaussian distribution* with standard deviation r and centered at \mathbf{c} , denoted by $\overline{D}_{\mathbb{Z}^n, r, \mathbf{c}}$, to be one that samples a polynomial according to the discrete Gaussian $D_{\mathbb{Z}^n, r, \mathbf{c}}$ and repeats the sampling if the polynomial is not $(r\sqrt{n})$ -bounded. As long as $n = \omega(\log(\kappa))$, Lemma 2.7 implies that this distribution is statistically close to the discrete Gaussian : $\overline{D}_{\mathbb{Z}^n, r, \mathbf{c}} \approx_s D_{\mathbb{Z}^n, r, \mathbf{c}}$.

The second fact says that the statistical distance between a discrete Gaussian with standard deviation r and centered at 0, and one centered at $\mathbf{c} \in \mathbb{Z}^n$ is at most $\|\mathbf{c}\|/r$. In particular, if r is super-polynomially larger than $\|\mathbf{c}\|$ then the two distributions are statistically close.

Lemma 2.8. *Let $n \in \mathbb{N}$. For any real number $r > \omega(\sqrt{\log n})$, and any $\mathbf{c} \in \mathbb{Z}^n$, the statistical distance between the distributions $D_{\mathbb{Z}^n, r}$ and $D_{\mathbb{Z}^n, r, \mathbf{c}}$ is at most $\|\mathbf{c}\|/r$.*

Corollary 2.9. *Let $\mathbf{c} \in \mathbb{Z}^n$. For any real number $r \geq 2^{\omega(\log \kappa)} \|\mathbf{c}\|$, the distributions $D_{\mathbb{Z}^n, r}$ and $D_{\mathbb{Z}^n, r, \mathbf{c}}$ are statistically close.*

2.6.2 The Ring LWE Assumption

We now describe the Ring Learning With Errors (RLWE) assumption of Lyubashevsky, Peikert, and Regev [LPR10]. The RLWE assumption is analogous to the standard Learning With Errors (LWE) assumption, first defined by Regev [Reg05, Reg09] (generalizing the learning parity with noise assumption of Blum et al. [BFKL93]).

The $\text{RLWE}_{\phi, q, \chi}$ assumption is that for a random ring element $s \leftarrow R_q$, given any polynomial number of samples of the form $(a_i, b_i = a_i \cdot s + e_i) \in R_q^2$, where a_i is uniformly random in R_q and e_i is drawn from the error distribution χ , the b_i 's are computationally indistinguishable from uniform in R_q . We use the *Hermite normal form* of the assumption, as in [BV11b], where the secret s is sampled from the noise distribution χ rather than being uniform in R_q . This presentation is more useful for the purposes of this work and is equivalent to the original up to obtaining one additional sample [ACPS09, LPR10].

Definition 2.13. (THE RLWE ASSUMPTION - HERMITE NORMAL FORM [LPR10]) *For all $\kappa \in \mathbb{N}$, let $\phi(x) = \phi_\kappa(x) \in \mathbb{Z}[x]$ be a polynomial of degree $n = n(\kappa)$, let $q = q(\kappa) \in \mathbb{Z}$ be an odd prime integer, let the ring $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$ and $R_q \stackrel{\text{def}}{=} R/qR$, and let χ denote a distribution over the ring R .*

The Decisional Ring LWE assumption $\text{RLWE}_{\phi, q, \chi}$ states that for any $\ell = \text{poly}(\kappa)$ it holds that

$$\{(a_i, a_i \cdot s + e_i)\}_{i \in [\ell]} \stackrel{c}{\approx} \{(a_i, u_i)\}_{i \in [\ell]},$$

where s is sampled from the noise distribution χ , a_i are uniform in R_q , the “error polynomials” e_i are sampled from the error distribution χ , and finally, the ring elements u_i are uniformly random over R_q .

We now present a couple of facts about the RLWE assumption. The first says that the assumption also holds if the error is multiplied by 2 in every sample. This follows immediately from the fact that q is an odd prime and therefore relatively prime with 2.

Fact 2.10. *The $\text{RLWE}_{\phi, q, \chi}$ assumption implies that for any $\ell = \text{poly}(\kappa)$,*

$$\{(a_i, a_i \cdot s + 2 \cdot e_i)\}_{i \in [\ell]} \stackrel{c}{\approx} \{(a_i, u_i)\}_{i \in [\ell]}.$$

where a_i, u_i are uniformly random in R_q and s, e_i are drawn from the error distribution χ .

The second fact says that the assumption also holds if the distinguisher is additionally given samples with the same parameter \mathbf{a}_j and different secret key s_i . This follows from a hybrid argument that slowly changes the samples, one secret at a time, from RLWE to uniform.

Fact 2.11. *The $\text{RLWE}_{\phi,q,\chi}$ assumption implies that for any $\ell = \text{poly}(\kappa), \ell' = \text{poly}(\kappa)$,*

$$\{(a_j, a_j \cdot s_i + e_{i,j})\}_{i \in [\ell], j \in [\ell']} \stackrel{c}{\approx} \{(a_j, u_{i,j})\}_{i \in [\ell], j \in [\ell']} .$$

where $a_j, u_{i,j}$ are uniformly random in R_q and $s_i, e_{i,j}$ are drawn from the error distribution χ .

2.6.3 Choice of Parameters

As already stated above, we will rely of the following specific choices of the polynomial $\phi(x)$ and the error distribution χ . For security parameter κ and a dimension parameter $n = n(\kappa)$ which is a power of two:

- We set $\phi(x) \stackrel{\text{def}}{=} x^n + 1$ where n is a power of two.
- The error distribution χ is the truncated discrete Gaussian distribution $\overline{D}_{\mathbb{Z}^n, r}$ with standard deviation $r > 0$. A sample from this distribution is a $(r\sqrt{n})$ -bounded polynomial $e \in R$.

2.6.4 The Worst-case to Average-case Connection

We state a worst-case to average-case reduction from the shortest vector problem on ideal lattices to the RLWE problem for our setting of parameters. The reduction stated below is a special case of the results of [LPR10].

Theorem 2.12 ([LPR10]). *Let $\phi(x) = x^n + 1$ where n is a power of two. Let $r \geq \omega(\sqrt{\log n})$ be a real number, and let $q \equiv 1 \pmod{2n}$ be a prime integer. Let $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$. Then there is a randomized reduction from $2^{\omega(\log n)} \cdot (q/r)$ -approximate R -SVP to $\text{RLWE}_{\phi,q,\chi}$ where $\chi = D_{\mathbb{Z}^n, r}$ is the discrete Gaussian distribution.*

Solving approximate R -SVP to within a sub-exponential factor is believed to be hard. Thus, if $q/r = 2^{o(n)}$ then the $\text{RLWE}_{\phi,q,\chi}$ assumption is believed to be hard.

2.7 NTRU Encryption

We describe the NTRU encryption scheme of Hoffstein et al. [HPS98], with the modifications proposed by Stehlé and Steinfeld [SS11b]. For security parameter κ , the scheme is parameterized by a prime number $q = q(\kappa)$, a degree $n = n(\kappa)$ polynomial $\phi(x) \in \mathbb{Z}[x]$, and an error distribution $\chi = \chi(\kappa)$ over the ring $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/\langle \phi(x) \rangle$. The parameters n, ϕ, q, χ are public and we assume that given κ , there are polynomial-time algorithms that output ϕ and q , and sample from the error distribution χ . The message space is $\mathcal{M} = \{0, 1\}$, and all operations are carried out in the ring R (i.e. modulo $\phi(x)$).

- **Keygen**(1^κ) : Sample polynomials $f', g \leftarrow \chi$ and set $f \stackrel{\text{def}}{=} 2f' + 1$ so that $f \equiv 1 \pmod{2}$. If f is not invertible in R_q , resample f' ; otherwise, let f^{-1} be the inverse of f in R_q . Set

$$\text{pk} \stackrel{\text{def}}{=} h := [2gf^{-1}]_q \in R_q \quad , \quad \text{sk} \stackrel{\text{def}}{=} f \in R$$

- $\text{Enc}(\text{pk}, m)$: To encrypt a bit $m \in \{0, 1\}$ with public key $\text{pk} = h$, sample polynomials $s, e \leftarrow \chi$ and output the ciphertext

$$c \stackrel{\text{def}}{=} [hs + 2e + m]_q \in R_q$$

- $\text{Dec}(\text{sk}, c)$: To decrypt a ciphertext $c \in R_q$ with secret key $\text{sk} = f$, let $\mu \stackrel{\text{def}}{=} [fc]_q$ and output $m \stackrel{\text{def}}{=} \mu \pmod{2}$.

It is easily seen that this scheme is correct as long as there is no reduction modulo q . To decrypt a ciphertext c , we compute:

$$[fc]_q = [fhs + 2fe + fm]_q = [2gs + 2fe + fm]_q$$

If there is no reduction modulo q then

$$[fc]_q \pmod{2} = 2gs + 2fe + fm \pmod{2} = fm \pmod{2} = m$$

Furthermore, our choice of parameter $\phi(x) = x^n + 1$ ensures there is no reduction modulo q . Notice that since the coefficients of g, s, e are all bounded by B , and the coefficients of f are bounded by $2B + 1$. By Corollary 2.5, we know that the coefficients of $[fc]_q$ are bounded by $2nB^2(2nB + 1)(2B + 1)$. As long as we set q to be large enough so that $q/2$ is larger than this quantity, a fresh ciphertext generated by Enc is guaranteed to decrypt correctly. From here on, we refer to $\mu = [fc]_q \in R_q$ as the “error in ciphertext c ”.

2.7.1 Security

The security of the (modified) NTRU encryption scheme can be based on two assumptions – the RLWE assumption described in Section 2.6, as well as an assumption that we call the (Decisional) Small Polynomial Ratio (DSPR) Assumption.

Definition 2.14. (DECISIONAL SMALL POLYNOMIAL RATIO ASSUMPTION) *Let $\phi(x) \in \mathbb{Z}[x]$ be a polynomial of degree n , let $q \in \mathbb{Z}$ be a prime integer, and let χ denote a distribution over the ring $R \stackrel{\text{def}}{=} \mathbb{Z}[x]/\langle\phi(x)\rangle$. The (decisional) small polynomial ratio assumption $\text{DSPR}_{\phi, q, \chi}$ says that it is hard to distinguish the following two distributions:*

- a polynomial $h \stackrel{\text{def}}{=} [2gf^{-1}]_q$, where f' and g are sampled from the distribution χ (conditioned on $f \stackrel{\text{def}}{=} 2f + 1$ being invertible over R_q) and f^{-1} is the inverse of f in R_q .
- a polynomial u sampled uniformly at random over R_q .

The security proof uses a hybrid argument, in two steps.

1. The hardness of $\text{DSPR}_{\phi, q, \chi}$ allows to change the public key $h = [2gf^{-1}]_q$ to a uniformly sampled h .
2. Once this is done, we can use $\text{RLWE}_{\phi, q, \chi}$ to change the challenge ciphertext $c^* = [hs + 2e + m]_q$ to $c^* = [u + m]_q$, where u is uniformly sampled from R_q .

In this final hybrid, the advantage of the adversary is exactly $1/2$ since c^* is uniform over R_q , independent of the message m .

Stehlé and Steinfeld [SS11b] showed that the $\text{DSPR}_{\phi,q,\chi}$ assumption is unconditionally true even for unbounded adversaries (namely, the two distributions above are statistically close) if n is a power of two, $\phi(x) = x^n + 1$, and χ is the discrete Gaussian $D_{\mathbb{Z}^n, r}$ for $r > \sqrt{q} \cdot \text{poly}(n)$. Thus, with this setting of parameters, semantic security of the modified NTRU scheme can be based on the $\text{RLWE}_{\phi,q,\chi}$ assumption alone.

3 Multikey FHE

As mentioned earlier, the main building block in our construction of on-the-fly MPC is multikey FHE: fully homomorphic encryption that allows homomorphic evaluation on ciphertexts encrypted under different and independent keys. In this chapter, we formally define multikey FHE and show a construction for any number of keys based on the NTRU encryption scheme [HPS98, SS11b] described in Section 2.7. We also show that any FHE scheme is inherently multikey for a constant number of keys (in the security parameter), and that the Brakerski-Vaikuntanathan scheme [BV11b, BGV12] is somewhat homomorphic for a logarithmic number of keys.

3.1 Definition

To formally define multikey fully homomorphic encryption, we introduce a parameter N , which is the number of distinct keys that the scheme can handle; all algorithms will depend polynomially on N . This is similar to the definition of leveled homomorphic encryption from [BGV12] (see Definition 2.7), but we note that in our definition, the algorithms depend on N but are independent of the depth of circuits that the scheme can evaluate. Thus, we consider schemes that are “leveled” with respect to the number of keys N , but fully homomorphic (“non-leveled”) with respect to the circuits that are evaluated. The construction of multikey FHE schemes that are not leveled with respect to the number of keys (i.e., where all algorithms are independent of N) remains an open problem.

Finally, we note that to guarantee semantic security, decryption requires all corresponding secret keys.

Definition 3.1 (Multikey \mathcal{C} -Homomorphic Encryption). *Let \mathcal{C} be a class of circuits. A family $\{\mathcal{E}^{(N)} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})\}_{N>0}$ of algorithms is multikey \mathcal{C} -homomorphic if for all integers $N > 0$, $\mathcal{E}^{(N)}$ has the following properties:*

- $(\text{pk}, \text{sk}, \text{ek}) \leftarrow \text{Keygen}(1^\kappa)$: For a security parameter κ , outputs a public key pk , a secret key sk and a (public) evaluation key ek .
- $c \leftarrow \text{Enc}(\text{pk}, m)$: Given a public key pk and message m , outputs a ciphertext c .
- $m := \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c)$: Given N secret keys $\text{sk}_1, \dots, \text{sk}_N$ and a ciphertext c , outputs a message m .
- $c := \text{Eval}(C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_\ell, \text{pk}_\ell, \text{ek}_\ell))$: Given a (description of) a boolean circuit C along with ℓ tuples $(c_i, \text{pk}_i, \text{ek}_i)$, each comprising of a ciphertext c_i , a public key pk_i , and an evaluation key ek_i , outputs a ciphertext c .

We require absence of decryption failures and compactness of ciphertexts. Formally: for every circuit $C \in \mathcal{C}$, all sequences of N key tuples $\{(\text{pk}'_j, \text{sk}'_j, \text{ek}'_j)\}_{j \in [N]}$ each of which is in

the support of $\text{Keygen}(1^\kappa)$, all sequences of ℓ key tuples $\{(\text{pk}_i, \text{sk}_i, \text{ek}_i)\}_{i \in [\ell]}$ each of which is in $\{(\text{pk}'_j, \text{sk}'_j, \text{ek}'_j)\}_{j \in [N]}$, and all plaintexts (m_1, \dots, m_ℓ) and ciphertexts (c_1, \dots, c_ℓ) such that c_i is in the support of $\text{Enc}(\text{pk}_i, m_i)$, Eval satisfies the following properties:

Correctness: Let $c := \text{Eval}(C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_\ell, \text{pk}_\ell, \text{ek}_\ell))$. Then $\text{Dec}(\text{sk}'_1, \dots, \text{sk}'_N, c) = C(m_1, \dots, m_\ell)$.⁶

Compactness: Let $c := \text{Eval}(C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_\ell, \text{pk}_\ell, \text{ek}_\ell))$. There exists a polynomial P such that $|c| \leq P(\kappa, N)$. In other words, the size of c is independent of ℓ and $|C|$. Note, however, that we allow the evaluated ciphertext to depend on the number of keys, N .

Definition 3.2 (Multikey Fully Homomorphic Encryption). A family of encryption schemes $\{\mathcal{E}^{(N)} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})\}_{N > 0}$ is multikey fully homomorphic if it is multikey \mathcal{C} -homomorphic for the class \mathcal{C} of all circuits.

Semantic security of a multikey FHE follows directly from the semantic security of the underlying encryption scheme in the presence of the evaluation key ek . This is because given ek , the adversary can compute Eval himself. Note that taking $N = 1$ in Definition 3.1 and Definition 3.2 yield the standard definitions of \mathcal{C} -homomorphic and fully homomorphic encryption schemes (Definition 2.5 and Definition 2.6).

3.2 Multikey FHE for a Small Number of Keys

As a prelude to our main result in Section 3.3, we show that multikey homomorphic encryption for a small number of keys can be easily achieved. In particular, we show that any (standard) FHE can be converted into a multikey FHE for a constant number of keys, $N = O(1)$. Furthermore, we show that the Brakerski-Vaikuntanathan (ring-based) FHE [BV11b] is multikey homomorphic for a logarithmic number of keys, $N = O(\log \kappa)$. Unfortunately, once we introduce multiple keys we are unable to use either relinearization or squashing, and can therefore only obtain a somewhat homomorphic encryption scheme.

3.2.1 $O(1)$ -Multikey FHE from any FHE

We show that any FHE scheme is inherently multikey for a constant number of keys, $N = O(1)$.⁷ Let $\mathcal{E} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})$ be an FHE scheme with message space $\{0, 1\}$ and ciphertext space $\{0, 1\}^\lambda$ where $\lambda = p(\kappa)$ for some polynomial $p(\cdot)$. For $x \in \{0, 1\}^*$, define $x[i]$ to be the i th bit of x , and define $\widetilde{\text{Enc}}$ to be the bit-wise encryption of x :

$$\widetilde{\text{Enc}}(\text{pk}, x) \stackrel{\text{def}}{=} (\text{Enc}(\text{pk}, x[1]), \dots, \text{Enc}(\text{pk}, x[|x|]))$$

⁶Note that correctness still holds even if the circuit C completely ignores all ciphertexts encrypted under a public key pk'_i , or if none of the original ciphertexts were encrypted under this key. In other words, using superfluous keys in the decryption process does not affect its correctness (as long as decryption uses at most N keys).

⁷The idea for this construction was originally suggested to us by an anonymous STOC 2012 reviewer. We include it in this dissertation and formally prove its correctness for the sake of completeness.

Furthermore, for any $k \in \mathbb{N}$, recursively define “onion” encryption and decryption:

$$\begin{aligned}
\text{Enc}^*(\text{pk}, x) &\stackrel{\text{def}}{=} \text{Enc}(\text{pk}, x) \\
\text{Enc}^*(\text{pk}_1, \dots, \text{pk}_k, x) &\stackrel{\text{def}}{=} \text{Enc}^*(\text{pk}_1, \dots, \text{pk}_{k-1}, \text{Enc}(\text{pk}_k, x)) \\
\text{Dec}^*(\text{sk}, x) &\stackrel{\text{def}}{=} \text{Dec}(\text{sk}_1, \text{Enc}(\text{pk}_2, \dots, \text{Enc}((\text{pk}_k, x)))) \\
\text{Dec}^*(\text{sk}_1, \dots, \text{sk}_k, x) &\stackrel{\text{def}}{=} \text{Dec}^*(\text{sk}_2, \dots, \text{pk}_k, \text{Dec}(\text{sk}_1, x)) \\
&= \text{Dec}(\text{sk}_k, \text{Dec}(\text{sk}_{k-1}, \dots, \text{Dec}(\text{sk}_1, x)))
\end{aligned}$$

We highlight two properties of “onion” encryption and decryption:

1. First, note that Enc^* and Dec^* satisfy correctness: if $(\text{pk}_i, \text{sk}_i) \leftarrow \text{Keygen}(1^\kappa)$ for all $i \in [k]$, then for all $m \in \{0, 1\}^\lambda$:

$$\text{Dec}^*(\text{sk}_1, \dots, \text{sk}_k, \text{Enc}^*(\text{pk}_1, \dots, \text{pk}_k, m)) = m$$

2. Second, note that the bit-size of the ciphertext $\text{Enc}^*(\text{pk}_1, \dots, \text{pk}_k, m)$ is λ^k . Recall that the ciphertext space of Enc is $\{0, 1\}^\lambda$ and $\lambda = p(\kappa)$ for some polynomial $p(\cdot)$.

Construction Overview. We now give an overview of the construction. Given N ciphertexts $c_i \leftarrow \text{Enc}(\text{pk}_i, m_i)$ encrypting plaintext m_i under key pk_i , for all $i \in [N]$, it is possible to homomorphically compute “onion” ciphertexts:

$$z_i \approx \text{Enc}^*(\text{pk}_1, \dots, \text{pk}_N, m_i)$$

This is done by homomorphically evaluating the function $\text{Enc}^*(\text{pk}_{i+1}, \dots, \text{pk}_N, \cdot)$ on ciphertext c_i . This outputs an onion encryption $\tilde{z}_i \approx \text{Enc}^*(\text{pk}_i, \dots, \text{pk}_N, m_i)$. The ciphertext z_i can be obtained by onion encrypting \tilde{z}_i with the remaining keys: $z_i = \text{Enc}^*(\text{pk}_1, \dots, \text{pk}_{i-1}, \tilde{z}_i)$

Once the ciphertexts z_1, \dots, z_N have been obtained, we can recursively perform homomorphic evaluations corresponding to the keys $\text{pk}_1, \dots, \text{pk}_N$ (in that order), to obtain a ciphertext:

$$c \approx \text{Enc}^*(\text{pk}_1, \dots, \text{pk}_N, C(m_1, \dots, m_N))$$

By correctness of “onion” encryption, decrypting c can be easily achieved using “onion” decryption:

$$\text{Dec}^*(\text{sk}_1, \dots, \text{sk}_N, c) = C(m_1, \dots, m_N)$$

However, recall that the size of each ciphertext z_i is $\lambda^N = p(\kappa)^N$ for some polynomial $p(\cdot)$. This means that the multikey homomorphic evaluation is efficient only if $N = O(1)$. Thus, this generic construction of multikey FHE from (standard) FHE allows only a constant number of keys.

Formal Description. We now give a formal description of the generic multikey construction, and prove its correctness. Let $\mathcal{E} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})$ be an FHE scheme with message space $\{0, 1\}^\lambda$ and ciphertext space $\{0, 1\}^\lambda$ where $\lambda = p(\kappa)$ for some polynomial $p(\cdot)$. Let Enc^* and Dec^* be the “onion” encryption and decryption algorithms described above.

- $\text{GMK.Keygen}(1^\kappa) : \text{Run Keygen}(1^\kappa)$.

- $\text{GMK.Enc}(\text{pk}, m) : \text{Run } \text{Enc}(\text{pk}, m).$
- $\text{GMK.Dec}(\text{sk}_1, \dots, \text{sk}_N, c) : \text{Output } \text{Dec}^*(\text{sk}_1, \dots, \text{sk}_N, c).$
- $\text{GMK.Eval}(C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_N, \text{pk}_N, \text{ek}_N)) : \text{For } i \in [N], \text{ define}$

$$G_i(x) \stackrel{\text{def}}{=} \text{Enc}^*(\text{pk}_{i+1}, \dots, \text{pk}_N, x; r)$$

for some fixed and valid randomness r ,⁸ and recursively define

$$C^{(k)}(x_1, \dots, x_N) \stackrel{\text{def}}{=} \begin{cases} C(x_1, \dots, x_N) & \text{for } k = N \\ \text{Eval}(\text{ek}_{k+1}, C^{(k+1)}, x_1, \dots, x_N) & \text{for } k < N \end{cases}$$

For $i \in [N]$, compute

$$\tilde{z}_i \stackrel{\text{def}}{=} \text{Eval}(\text{ek}_i, G_i, c_i) \quad , \quad z_i \stackrel{\text{def}}{=} \text{Enc}^*(\text{pk}_1, \dots, \text{pk}_{i-1}, \tilde{z}_i)$$

and output the ciphertext $c \stackrel{\text{def}}{=} \text{Eval}(\text{ek}_1, C^{(1)}, z_1, \dots, z_N).$

Theorem 3.1. *The encryption scheme $\mathcal{E}_{\text{GMK}} = (\text{GMK.Keygen}, \text{GMK.Enc}, \text{GMK.Dec}, \text{GMK.Eval})$ is multikey fully homomorphic for $N = O(1)$ keys.*

Proof. To prove correctness of evaluation, we wish to prove that if $(\text{pk}_i, \text{sk}_i, \text{ek}_i)$ is in the support of $\text{GMK.Keygen}(1^\kappa) = \text{Keygen}(1^\kappa)$ and $c_i \leftarrow \text{GMK.Enc}(\text{pk}_i, m_i) = \text{Enc}(\text{pk}_i, m_i)$, then

$$\text{GMK.Dec}(\text{sk}_1, \dots, \text{sk}_N, c) = \text{Dec}^*(\text{sk}_1, \dots, \text{sk}_N, c) = C(m_1, \dots, m_N)$$

We first show that each z_i is a valid “onion” encryption of m_i . By correctness of evaluation with evaluation key ek_i , we know that

$$\text{Dec}(\text{sk}_i, \tilde{z}_i) = G_i(m_i) = \text{Enc}^*(\text{pk}_{i+1}, \dots, \text{pk}_N, m_i; r)$$

and by correctness of encryption, we conclude that

$$\text{Dec}^*(\text{sk}_1, \dots, \text{sk}_N, \tilde{z}_i) = m_i \quad \text{and} \quad \text{Dec}^*(\text{sk}_1, \dots, \text{sk}_N, z_i) = m_i$$

We now make the following claim, which constitutes the bulk of the proof.

Claim 3.1.1. *For every $k \in [N]$,*

$$\text{Dec}^*(\text{sk}_1, \dots, \text{sk}_k, c) = C_k(z_1^{(k)}, \dots, z_N^{(k)})$$

where $z_i^{(k)} \stackrel{\text{def}}{=} \text{Dec}^*(\text{sk}_1, \dots, \text{sk}_k, z_i).$

In particular, for $k = N$, this claim implies:

$$\text{Dec}^*(\text{sk}_1, \dots, \text{sk}_N, c) = C^{(N)}(z_1^{(N)}, \dots, z_N^{(N)}) = C(m_1, \dots, m_N)$$

where the second equality follows from the fact that $C_N = C$ by definition, and the fact that $z_i^{(N)} = \text{Dec}^*(\text{sk}_1, \dots, \text{sk}_N, z_i) = m_i$, which we proved earlier.

It thus suffices to prove Claim 3.1.1 to conclude the proof of the theorem.

⁸We need to include the randomness in the definition because we want $G_i(x)$ to be a deterministic circuit with x as its sole input.

Proof. We prove Claim 3.1.1 by induction. The base case, $k = 1$, follows directly from correctness of evaluation and correctness of decryption:

$$\text{Dec}^*(\text{sk}_1, c) = C^{(1)}(\text{Dec}(\text{sk}_1, z_1), \dots, \text{Dec}(\text{sk}_1, z_N)) = C^{(1)}(z_1^{(1)}, \dots, z_N^{(1)})$$

Now suppose that the claim holds for $k - 1$; that is, suppose

$$\text{Dec}^*(\text{sk}_1, \dots, \text{sk}_{k-1}, c) = C^{(k-1)}(z_1^{(k-1)}, \dots, z_N^{(k-1)})$$

Decrypting both sides by sk_k yields:

$$\begin{aligned} \text{Dec}^*(\text{sk}_1, \dots, \text{sk}_k, c) &= \text{Dec}\left(\text{sk}_k, C^{(k-1)}(z_1^{(k-1)}, \dots, z_N^{(k-1)})\right) \\ &= \text{Dec}\left(\text{sk}_k, \text{Eval}\left(\text{ek}_k, C^{(k)}, z_1^{(k-1)}, \dots, z_N^{(k-1)}\right)\right) \\ &= C^{(k)}\left(\text{Dec}\left(\text{sk}_k, z_1^{(k-1)}\right), \dots, \text{Dec}\left(\text{sk}_k, z_N^{(k-1)}\right)\right) \\ &= C^{(k)}(z_1^{(k)}, \dots, z_N^{(k)}) \end{aligned}$$

where the second-to-last equality follows from correctness of evaluation and correctness of decryption. This concludes the inductive step and the proof. \square

\square

3.2.2 $O(\log \kappa)$ -Multikey FHE from Ring-LWE

We now show that the Brakerski-Vaikuntanathan FHE [BV11b] based on the RLWE assumption is multikey somewhat homomorphic for $N = O(\log \kappa)$ keys.

Decryption in Regev-style encryption consists of computing the inner product $\langle \mathbf{c}, \mathbf{s} \rangle \pmod{2}$, where $\mathbf{c}, \mathbf{s} \in R_q^2$ are the ciphertext and secret key, respectively. Brakerski and Vaikuntanathan [BV11b] generalize this to allow the ciphertext and secret key to grow in dimension. For $\mathbf{c}, \mathbf{s} \in R_q^d$, they define: $\text{Dec}(\mathbf{s}, \mathbf{c}) = \langle \mathbf{c}, \mathbf{s} \rangle \pmod{2}$. Homomorphic operations are then defined as follows:

- Given two *same-length* ciphertexts \mathbf{c}_1 and \mathbf{c}_2 , output the ciphertext $\mathbf{c}_{\text{add}} \stackrel{\text{def}}{=} \mathbf{c}_1 + \mathbf{c}_2$ as an encryption of the *sum* of the underlying messages.

The ciphertext \mathbf{c}_{add} is decryptable with the same secret key \mathbf{s} since

$$\langle \mathbf{c}_1 + \mathbf{c}_2, \mathbf{s} \rangle = \langle \mathbf{c}_1, \mathbf{s} \rangle + \langle \mathbf{c}_2, \mathbf{s} \rangle$$

- Given two ciphertexts \mathbf{c}_1 and \mathbf{c}_2 of *potentially different length*, output the ciphertext $\mathbf{c}_{\text{mult}} \stackrel{\text{def}}{=} \mathbf{c}_1 \otimes \mathbf{c}_2$ as the *product* of the underlying messages.

The ciphertext \mathbf{c}_{mult} is now decryptable with the secret key $\mathbf{s} \otimes \mathbf{s}$ since

$$\langle \mathbf{c}_1 \otimes \mathbf{c}_2, \mathbf{s} \otimes \mathbf{s} \rangle = \langle \mathbf{c}_1, \mathbf{s} \rangle \cdot \langle \mathbf{c}_2, \mathbf{s} \rangle$$

We can extend this to the multikey setting. Multiplication is trivial, but some changes are necessary in the case of addition.

- Given two *same-length* ciphertexts \mathbf{c}_1 and \mathbf{c}_2 decryptable with secret keys $\mathbf{s}_1, \mathbf{s}_2$ respectively, output the ciphertext $\mathbf{c}_{\text{add}} \stackrel{\text{def}}{=} (\mathbf{c}_1, \mathbf{c}_2)$ as an encryption of the *sum* of the underlying messages. The ciphertext \mathbf{c}_{add} is decryptable with the same secret key $(\mathbf{s}_1, \mathbf{s}_2)$ since

$$\langle (\mathbf{c}_1, \mathbf{c}_2), (\mathbf{s}_1, \mathbf{s}_2) \rangle = \langle \mathbf{c}_1, \mathbf{s}_1 \rangle + \langle \mathbf{c}_2, \mathbf{s}_2 \rangle$$

- Given two ciphertexts \mathbf{c}_1 and \mathbf{c}_2 decryptable with secret keys $\mathbf{s}_1, \mathbf{s}_2$ respectively, and *of potentially different length*, output the ciphertext $\mathbf{c}_{\text{mult}} \stackrel{\text{def}}{=} \mathbf{c}_1 \otimes \mathbf{c}_2$ as the *product* of the underlying messages.

The ciphertext \mathbf{c}_{mult} is now decryptable with the secret key $\mathbf{s}_1 \otimes \mathbf{s}_2$ since

$$\langle \mathbf{c}_1 \otimes \mathbf{c}_2, \mathbf{s}_1 \otimes \mathbf{s}_2 \rangle = \langle \mathbf{c}_1, \mathbf{s}_1 \rangle \cdot \langle \mathbf{c}_2, \mathbf{s}_2 \rangle$$

Observe that each homomorphic operation (at most) doubles the size of the ciphertext. Starting with fresh ciphertexts of length 2, after $(N-1)$ operations (which can combine ciphertexts encrypted under at most N distinct keys), the size of both the ciphertext and the joint decryption key is 2^N . This is only feasible if $N = O(\log \kappa)$.

As shown in the work of Brakerski and Vaikuntanathan [BV11b], the scheme can evaluate circuits of depth $D < \varepsilon \log n - \log \log n + \Theta(1)$, where $q = 2^{n^\varepsilon}$ for constant $\varepsilon \in (0, 1)$. Unfortunately, we do not know how to apply relinearization or squashing in the multikey setting, and are therefore not able to convert the resulting multikey scheme into a leveled or fully homomorphic one.

3.3 Multikey Somewhat Homomorphic Encryption for Any Number of Keys

We now turn to our main result in this section: we construct a multikey somewhat homomorphic encryption scheme based on the (modified) NTRU encryption scheme [HPS98, SS11b] described in Section 2.7. Unlike the schemes in Section 3.2, the scheme we describe in this section will be multikey for $N \approx n^\varepsilon$ keys, with constant $\varepsilon \in (0, 1)$. In Section 3.4, we show how to convert the scheme into a multikey fully homomorphic scheme for $N \approx n^\varepsilon$ keys. By setting $n \approx N^{1/\varepsilon}$, we can construct a multikey FHE for any number of keys N , as long as N is known a-priori.

We begin by informally describing the multikey homomorphic properties of NTRU encryption and some of the problems encountered when trying to convert the scheme from Section 2.7 into a somewhat homomorphic one. We then show a formal description of our somewhat homomorphic scheme, formally prove its homomorphic properties, and discuss its security. In Section 3.4, we show how to convert this scheme into a fully homomorphic scheme.

3.3.1 Multikey Homomorphism

Recall from Section 2.7 that an NTRU key pair consists of ring elements (h, f) such that $h = [2gf^{-1}]_q$, where g, f are “small” ring elements sampled from a B -bounded distribution χ , and f^{-1} is the inverse of f in R_q . Further recall that an NTRU ciphertext has the form $c = [hs + 2e + m]_q$ for “small” elements s, e sampled from χ , and decryption computes $[fc]_q \pmod{2}$.

Let (h_1, f_1) and (h_2, f_2) be two different NTRU public/secret key pairs, and let $c_1 \stackrel{\text{def}}{=} [h_1 s_1 + 2e_1 + m_1]_q$ and $c_2 \stackrel{\text{def}}{=} [h_2 s_2 + 2e_2 + m_2]_q$ be two ciphertexts, encrypted under public keys h_1 and h_2 , respectively. We show how to compute ciphertexts that decrypt to the sum and

the product of the underlying plaintexts, m_1 and m_2 . In particular, we show that the “ciphertexts” $c_{\text{mult}} \stackrel{\text{def}}{=} c_1 \cdot c_2$ and $c_{\text{add}} \stackrel{\text{def}}{=} c_1 + c_2$ can be decrypted to the product and the sum of m_1 and m_2 respectively, using the secret key $f_{12} \stackrel{\text{def}}{=} f_1 \cdot f_2$.

To see this, note that

$$\begin{aligned} [f_1 f_2 (c_1 + c_2)]_q &= [2f_1 f_2 e_1 + 2f_1 f_2 e_2 + 2f_2 g_1 s_1 + 2f_1 g_2 s_2 + f_1 f_2 (m_1 + m_2)]_q \\ [f_1 f_2 (c_1 \cdot c_2)]_q &= [4g_1 g_2 s_1 s_2 + 2g_1 s_1 f_2 (2e_2 + m_2) + 2g_2 s_2 f_1 (2e_1 + m_1) + \\ &\quad 2f_1 f_2 (e_1 m_2 + e_2 m_1 + 2e_1 e_2) + f_1 f_2 (m_1 m_2)]_q \\ &= m_1 \cdot m_2 \pmod{2} \end{aligned}$$

Since $f_1 \equiv f_2 \equiv 1 \pmod{2}$, we can conclude that as long as there is no reduction modulo q ,

$$\begin{aligned} [f_1 f_2 (c_1 + c_2)]_q \pmod{2} &= m_1 + m_2 \pmod{2} \\ [f_1 f_2 (c_1 \cdot c_2)]_q \pmod{2} &= m_1 \cdot m_2 \pmod{2} \end{aligned}$$

In other words, the “joint secret key” $f_{12} \stackrel{\text{def}}{=} f_1 f_2$ can be used to decrypt $c_{\text{add}} = [c_1 + c_2]_q$ and $c_{\text{mult}} = [c_1 \cdot c_2]_q$. We can extend this argument to any combination of operations, with ciphertexts encrypted under multiple public keys.

Of course, the error in the ciphertexts will grow with the number of operations performed (as with all known fully homomorphic encryption schemes). Thus, correctness of decryption will only hold for a limited number of operations. We can show that the scheme can correctly evaluate circuits of depth roughly $\varepsilon \log(n)$ when $q = 2^{n^\varepsilon}$ and $B = \text{poly}(n)$.

Problems in Multikey Decryption. An astute reader will have observed that in order to successfully decrypt a ciphertext that was the result of a homomorphic evaluation, we must know the circuit that was evaluated. For example, to decrypt $c_1^2 + c_2$ we need to multiply by $f_1^2 f_2$, whereas to decrypt $c_1 + c_2^2$ we need to multiply by $f_1 f_2^2$. This is unsatisfactory.

Furthermore, consider what happens when we add or multiply two ciphertexts c, c' that are themselves a result of homomorphic evaluation. Suppose, for example, that $c = c_1 c_2$ and $c' = c_2 c_3$, where c_i is encrypted under h_i for $i \in \{1, 2, 3\}$. We know c can be decrypted using $f_1 f_2$ and c' can be decrypted using $f_2 f_3$. Thus, we know that

$$[f_1 f_2 \cdot c]_q = 2e + f_1 f_2 m \quad , \quad [f_2 f_3 \cdot c']_q = 2e' + f_2 f_3 m'$$

for some messages m and m' and error terms e and e' . Following the discussion above, we can see that $c + c'$ can be decrypted using the key $f_1 f_2 f_3$:

$$[f_1 f_2 f_3 \cdot (c + c')]_q = [f_3 (f_1 f_2 \cdot c) + f_1 (f_2 f_3 \cdot c')]_q = 2(f_3 e + f_1 e') + f_1 f_2 f_3 (m + m')$$

In general, given a ciphertext c encrypted under a set of keys K , and c' encrypted under a set of keys K' , their sum can be decrypted using the product of the keys in the union $K \cup K'$. We note that the absolute magnitude of the coefficients of this product grows exponentially with the number of keys in $K \cup K'$, i.e. the total number of keys involved in the homomorphic computation.

Analogously, in the context of homomorphic multiplication, we need $f_1 f_2^2 f_3$ to decrypt $c \cdot c'$:

$$[f_1 f_2^2 f_3 \cdot (c \cdot c')]_q = [(f_1 f_2 \cdot c) \cdot (f_2 f_3 \cdot c')]_q = 2E_{\text{mult}} + f_1 f_2^2 f_3 (m \cdot m')$$

where $E_{\text{mult}} \stackrel{\text{def}}{=} 2ee' + ef_2 f_3 m' + e' f_1 f_2 m$. This hints at the fact that the magnitude of the coefficients of the joint secret key needed to decrypt an evaluated ciphertext grows *exponentially with the degree of the evaluated circuit* (and not just with the number of keys involved, as in the case of addition). Indeed, after M multiplications, the joint secret key needed to decrypt the evaluated ciphertext will be the product of M polynomials, and the magnitude of the coefficients of this product will be exponential in M .

Our Solution. To solve the above problems, we use *relinearization* (also known as *key-switching*), a technique first introduced by Brakerski and Vaikuntanathan [BV11a]. Informally, we introduce a (public) evaluation key ek that will be output by the Keygen algorithm. Every time we multiply ciphertexts that share a key f_i , we will use the evaluation key to ensure that we only need f_i , and not f_i^2 , to decrypt the new ciphertext. This ensures two things.

1. First, it ensures that to decrypt a ciphertext c^* , we only need to multiply it by *one* copy of each secret key, making decryption independent of the circuit used to produce c^* .
2. Second, it ensures that the size of the joint secret key needed to decrypt the new ciphertext depends only on the number of keys N , and not on the degree of the circuit C that was evaluated.

Though we are able to eliminate the dependence (of the magnitude of the coefficients of the joint secret key) on the degree of the circuit, we remark that we do not succeed in eliminating the exponential dependence on N , the number of keys – indeed, this is the reason why our solution will eventually assume an a-priori upper bound on N .

3.3.2 Formal Description

We present a formal description of our multikey somewhat homomorphic encryption scheme based on the (modified) NTRU encryption scheme [HPS98, SS11b] described in Section 2.7.

- $\text{SH.Keygen}(1^\kappa)$: Sample $f', g \leftarrow \chi$ and set $f := 2f' + 1$ so that $f \equiv 1 \pmod{2}$. If f is not invertible in R_q , resample f' ; otherwise let f^{-1} be the inverse of f in R_q . Set

$$\text{pk} \stackrel{\text{def}}{=} h := [2gf^{-1}]_q \in R_q \quad , \quad \text{sk} \stackrel{\text{def}}{=} f \in R$$

Sample $\tilde{s}, \tilde{e} \leftarrow \chi^{\lceil \log q \rceil}$ and compute $\text{ek} \stackrel{\text{def}}{=} [h\tilde{s} + 2\tilde{e} + \text{Pow}(f)]_q \in R_q^{\lceil \log q \rceil}$. Output the key tuple $(\text{pk}, \text{sk}, \text{ek})$.

- $\text{SH.Enc}(\text{pk}, m)$: Sample $s, e \leftarrow \chi$. Output the ciphertext $c := hs + 2e + m \in R_q$.
- $\text{SH.Dec}(\text{sk}_1, \dots, \text{sk}_N, c)$: Parse $\text{sk}_i = f_i$ for $i \in [N]$. Compute $\mu = [f_1 \cdots f_N \cdot c]_q \in R_q$ and output $m := \mu \pmod{2}$.

- **SH.Eval**($C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_\ell, \text{pk}_\ell, \text{ek}_\ell)$): We show how to evaluate an ℓ -variate boolean circuit $C : \{0, 1\}^\ell \rightarrow \{0, 1\}$ of depth D . To this end, we show how to homomorphically add and multiply two elements in $\{0, 1\}$. Given two ciphertexts c, c' , we assume that we also have a set of distinct public keys associated with each ciphertext.⁹ We will denote these lists by K, K' , respectively. The public-key set of a fresh encryption is simply the set $\{\text{pk}\}$ containing the public key under which it was encrypted. For convenience, in our analysis we sometimes assume that the set contains the indices of the public keys instead of the keys themselves.

- Given two ciphertexts c and c' with corresponding public-key sets K and K' , output the ciphertext

$$c_{\text{add}} = [c + c']_q \in R_q$$

as an encryption of the *sum* of the underlying messages. Output the set $K_{\text{add}} = K \cup K'$ as its corresponding public-key set.

- Given two ciphertexts c and c' with corresponding public-key sets K and K' , compute ciphertext $c_0 = [c \cdot c']_q \in R_q$.
 - * If $K \cap K' = \emptyset$, let $c_{\text{mult}} = c_0$.
 - * Otherwise, let $K \cap K' = \{\text{pk}_{i_1}, \dots, \text{pk}_{i_t}\}$. For $j \in [t]$, compute $c_j = [\langle \text{Bit}(c_{j-1}), \text{ek}_{i_j} \rangle]_q$, and let $c_{\text{mult}} = c_t$ at the end of the iteration.

In either case, output c_{mult} as an encryption of the *product* of the underlying messages, and output the set $K_{\text{mult}} = K \cup K'$ as its corresponding public-key set.

For a set $S \subseteq [N]$, let $f_S \stackrel{\text{def}}{=} \prod_{i \in S} f_i$. Note that the ciphertext c_0 can be decrypted to $m \cdot m'$ with the “joint” secret key $f_K f_{K'}$, which contains the term $f_{i_1}^2 \dots f_{i_t}^2$. The goal of relinearization is to convert it into a ciphertext that decrypts to the same message under the secret key

$$f_K f_{K'} \left(\prod_{j \in K \cap K'} f_j \right)^{-1} = f_{K \cup K'}$$

which replaces the term $f_{i_1}^2 \dots f_{i_t}^2$ with the term $f_{i_1} \dots f_{i_t}$.

We first show that the scheme works correctly as advertised:

Lemma 3.2. *If $q = 2^{n^\varepsilon}$ for $\varepsilon \in (0, 1)$ and χ is a B -bounded distribution for $B = \text{poly}(n)$, then the encryption scheme $\mathcal{E}_{\text{SH}} = (\text{SH.Keygen}, \text{SH.Enc}, \text{SH.Dec}, \text{SH.Eval})$ described above is multikey homomorphic for $N = O(n^\delta)$ keys and circuits of depth $D < (\varepsilon - \delta) \log n - \log \log n - O(1)$.*

Proof. We define the (multikey) error of a ciphertext c with corresponding public-key set K to be $\mu \stackrel{\text{def}}{=} [f_K \cdot c]_q$. We start by showing that the magnitude of the error coefficients does not grow too much after a homomorphic evaluation.

Claim 3.2.1. *Let c, c' be ciphertexts encrypting m and m' , respectively, and suppose that the magnitude of their error coefficients is bounded by $E < q/2$. Then c_{add} and c_{mult} correctly decrypt to $m + m'$ and $m \cdot m'$, respectively, and their error coefficients are bounded by $(nB)^{2N} E^2$.*

⁹That is, we assume each set contains distinct public keys, but the intersection of any two sets might not be empty.

Proof. Let c, c' be encryptions of m, m' , respectively, with corresponding public-key sets K, K' . We know that for some $e, e' \in R$ we have:

$$[f_K \cdot c]_q = 2e + m \quad , \quad [f_{K'} \cdot c']_q = 2e' + m'$$

and $\|2e + m\|_\infty, \|2e' + m'\|_\infty < E$. Then

$$\begin{aligned} [f_{K_{\text{add}}} \cdot c_{\text{add}}]_q &= [f_{K \cup K'} \cdot (c + c')]_q = [f_{K \setminus K'}(f_K \cdot c) + f_{K' \setminus K}(f_{K'} \cdot c')]_q \\ &= f_{K \setminus K'}(2e + m) + f_{K' \setminus K}(2e' + m') \end{aligned}$$

We can thus bound the magnitude of the coefficients of $[f_{K_{\text{add}}} \cdot c_{\text{add}}]_q$ by $2(nB)^N E < (nB)^{2N} E^2$, as desired. Furthermore, it is easy to see that $[f_{K_{\text{add}}} \cdot c_{\text{add}}]_q \pmod{2} = m + m'$.

The multiplication case is more complex. Let $K \cap K' = \{i_1, \dots, i_t\}$, as before. Define $F_0 \stackrel{\text{def}}{=} f_K f_{K'}$, and for $j \in [t]$, define $F_j = F_{j-1} \cdot f_{i_j}^{-1}$. Then $F_r = f_{K \cup K'}$ is a simple product of the secret keys f_i , without any quadratic terms. We know that

$$[F_0 \cdot c_0]_q = [(f_K \cdot c)(f_{K'} \cdot c_K)]_q = (2e + m)(2e' + m')$$

so that $\|[F_0 \cdot c_0]_q\|_\infty < nE^2$ and $[F_0 \cdot c_0]_q \pmod{2} = m \cdot m'$. Furthermore, for all $j \in [t]$,

$$\begin{aligned} [F_j \cdot c_j]_q &= [F_j \cdot \langle \text{Bit}(c_{j-1}), h_{i_j} \tilde{s} + 2\tilde{e} + \text{Pow}(f_{i_j}) \rangle]_q \\ &= [F_j \cdot \langle \text{Bit}(c_{j-1}), h_{i_j} \tilde{s} \rangle + F_j \cdot \langle \text{Bit}(c_{j-1}), 2\tilde{e} \rangle + F_j c_{j-1} f_{i_j}]_q \\ &= F_j f_{i_j}^{-1} \cdot \langle \text{Bit}(c_{j-1}), 2g_{i_j} \tilde{s} \rangle + F_j \cdot \langle \text{Bit}(c_{j-1}), 2\tilde{e} \rangle + F_{j-1} c_{j-1} \end{aligned}$$

Using the fact that each F_j is the product of at most $(2N - j)$ keys, we have that

$$\begin{aligned} \|[F_j \cdot c_j]_q\|_\infty &< 2 \lceil \log q \rceil n^2 B^2 \cdot (nB)^{2N-j-1} + 2 \lceil \log q \rceil nB \cdot (nB)^{2N-j} + \|[F_{j-1} \cdot c_{j-1}]_q\|_\infty \\ &= 4 \lceil \log q \rceil (nB)^{2N-j+1} + \|[F_{j-1} \cdot c_{j-1}]_q\|_\infty \end{aligned}$$

This yields the following bound on the error of c_{mult} :

$$\begin{aligned} \|[F_{K \cup K'} \cdot c_{\text{mult}}]_q\|_\infty &= \|[F_t \cdot c_t]_q\|_\infty \leq nE^2 + \sum_{j=1}^t 4 \lceil \log q \rceil (nB)^{2N-j+1} \\ &= nE^2 + 4 \lceil \log q \rceil (nB)^{2N+1} \sum_{j=1}^t (nB)^{-j} \\ &\leq nE^2 + 8 \lceil \log q \rceil (nB)^{2N+1} \\ &\leq (nB)^{2N} E^2 \end{aligned}$$

where the last inequality holds by the fact that $q = 2^{n^\varepsilon}$.

Furthermore, notice that $[F_j \cdot c_j]_q \equiv F_{j-1} c_{j-1} \pmod{2}$. Since $[F_0 \cdot c_0]_q \pmod{2} = m \cdot m'$, we can conclude that $[F_{K \cup K'} \cdot c_{\text{mult}}]_q \pmod{2} = [F_t \cdot c_t]_q \pmod{2} = m \cdot m'$. □

Once we have bounded the magnitude of the error coefficients after a homomorphic operation, we can bound the overall error incurred after evaluating a circuit of depth D . Starting with error $E_0 \leq 3(nB)^2$, after D levels of homomorphic operations, the error magnitude can grow to at most:

$$((nB)^{2N} E_0)^{2^D} \leq \left((3nB)^{2^D \cdot (2N+2)} \right)$$

This results in correct decryption as long as $D < \log \log q - \log \log n - \log N - O(1)$, where we use the fact that $B = \text{poly}(n)$. In particular, for $N = O(n^\delta)$ keys and $q = 2^{n^\varepsilon}$, we get $D < (\varepsilon - \delta) \log n - \log \log n - O(1)$. □

3.3.3 Security

Recall from Section 2.7 that the security of the (modified) NTRU encryption scheme can be based on two assumptions – the RLWE assumption and the DSPR assumption. Recall further that Stehlé and Steinfeld [SS11b] showed that the $\text{DSPR}_{\phi,q,\chi}$ assumption is unconditionally true if n is a power of 2, $\phi(x) = x^n + 1$ is the n^{th} cyclotomic polynomial, and χ is the discrete Gaussian $D_{\mathbb{Z}^n,r}$ for $r > \sqrt{q} \cdot \text{poly}(n)$. This allowed them to prove semantic security for the modified NTRU scheme under the $\text{RLWE}_{\phi,q,\chi}$ assumption alone.

Unfortunately, their result holds only if $r > \sqrt{q} \cdot \text{poly}(n)$, which is too large to permit even a single homomorphic multiplication. To support homomorphic operations, we need to use a much smaller value of r , for which it is easy to see that the $\text{DSPR}_{\phi,q,\chi}$ assumption does not hold in a statistical sense any more. Thus, it is necessary to assume that the decisional small polynomial ratio problem is hard for our choice of parameters.

Additionally, note that the evaluation key ek contains elements of the form $[hs_\tau + 2e_\tau + 2^\tau f]_q$, which can be thought of as “pseudo-encryptions” of (multiples of) the secret key f under the corresponding public key h .¹⁰ The security of the scheme must then additionally rely on a “circular security” assumption that states that semantic security of the scheme is maintained given the evaluation key as constructed above. We remark that this assumption can be avoided at the cost of obtaining a leveled homomorphic encryption scheme (where the size of the evaluation key grows with the depth of the circuits that the scheme supports).

Thus, we can base the security of the scheme on the DSPR assumption, the RLWE assumption, and the “circular security” assumption described above.

Lemma 3.3. *Let n be a power of 2, let $\phi(x) = x^n + 1$, let $q = 2^{n^\varepsilon}$ for $\varepsilon \in (0, 1)$ and $\chi = D_{\mathbb{Z}^n,r}$ with $r = \text{poly}(n)$. Then the somewhat homomorphic encryption scheme $\mathcal{E}_{\text{SH}} = \{\text{SH.Keygen}, \text{SH.Enc}, \text{SH.Dec}, \text{SH.Eval}\}$ described above is secure under the $\text{DSPR}_{\phi,q,\chi}$ and $\text{RLWE}_{\phi,q,\chi}$ assumptions, and the assumption that the scheme remains semantically secure even given the evaluation key ek .*

3.4 From Somewhat to Fully Homomorphic Encryption

We use a generalization of Gentry’s bootstrapping theorem [Gen09b, Gen09a] (see Section 2.5) to convert the multikey somewhat homomorphic scheme from Section 3.3 into a multikey fully homomorphic one. We modify Gentry’s bootstrapping theorem and the corresponding definitions from their original presentation to generalize them to the multikey setting.

¹⁰We point out that these are not true encryptions of the “message” $2^\tau f$ since $2^\tau f$ is not a binary polynomial, whereas our scheme is only equipped to correctly encrypt polynomials $m \in R_2$.

Definition 3.3 (Multikey Bootstrappable Schemes). Let $\mathcal{E} = \{\mathcal{E}^{(N)} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})\}_{N \geq 0}$ be a family of multikey \mathcal{C} -homomorphic encryption schemes, and let f_{add} and f_{mult} be the augmented decryption functions of the scheme defined as

$$\begin{aligned} f_{\text{add}}^{c_1, c_2}(\text{sk}_1, \dots, \text{sk}_N) &= \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c_1) \quad \text{XOR} \quad \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c_2) \\ f_{\text{mult}}^{c_1, c_2}(\text{sk}_1, \dots, \text{sk}_N) &= \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c_1) \quad \text{AND} \quad \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c_2) \end{aligned}$$

Then \mathcal{E} is bootstrappable if $\{f_{\text{add}}^{c_1, c_2}, f_{\text{mult}}^{c_1, c_2}\}_{c_1, c_2} \subseteq \mathcal{C}$. Namely, the scheme can homomorphically evaluate f_{add} and f_{mult} .

We now state a generalization of Gentry’s bootstrapping theorem to the multikey setting. Taking $N = 1$ yields the theorem and the definitions from [Gen09b, Gen09a] and Section 2.5.

Theorem 3.4 (Multikey Bootstrapping Theorem). Let \mathcal{E} be a bootstrappable family of multikey homomorphic schemes that is also weakly circular secure. Then there is a multikey fully homomorphic family of encryption schemes \mathcal{E}' .

Unfortunately, the somewhat homomorphic scheme described in Section 3.3 is not bootstrappable. Recall that the scheme can only evaluate circuits of depth less than $\varepsilon \log(n)$, where $\varepsilon < 1$. However, the shallowest implementation of the decryption circuit we are aware of has depth $c \log N \cdot \log n$ for some constant $c > 1$. We therefore turn to modulus reduction, a technique introduced by [BV11a] and refined by [BGV12], to convert our somewhat homomorphic scheme into a bootstrappable scheme.

3.4.1 Modulus Reduction

Modulus reduction [BV11a, BGV12] is a noise-management technique that provides an *exponential* gain on the depth of the circuits that can be evaluated, while keeping the depth of the decryption circuit unchanged. Informally, if D_{dec} is the depth of the decryption circuit of the multikey scheme described in Section 3.3.1, then we modify the scheme so that its decryption circuit is unchanged but the scheme can now evaluate circuits of depth D_{dec} . Hence, the new scheme can evaluate its own decryption circuit, making it bootstrappable. Details follow.

Let (h, f) be a key pair and let c be a ciphertext under public key h . Recall that for decryption to be successful, we need the error $[fc]_q$ to be equal to $fc \in R$. However, each homomorphic operation increases this error. Modulus reduction allows us to keep the error magnitude small by simply scaling the ciphertext after each operation. The key idea is to exploit the difference in how the error affects security and homomorphism:

- The growth of error upon homomorphic multiplication is governed by the *magnitude* of the noise.
- Security is governed by the *ratio* between the magnitude of the error and the modulus q .

This suggests a method of reducing the magnitude of the error and the modulus by roughly the same factor, thus preserving security while at the same time making homomorphic multiplications “easier”. In particular, modulus reduction gives us a way to transform a ciphertext $c \in R_q$ into a different ciphertext $c' \in R_p$ (for $p < q$) while preserving correctness: for “joint” secret key $f = \prod_{i=1}^N f_i$,

$$[fc]_p = [fc']_q \pmod{2}$$

The transformation from c to c' involves simply scaling by (p/q) and rounding appropriately.

Lemma 3.5 ([BGV12]). *Let p and q be two odd moduli, and let $c \in R_q$. Define c' to be the polynomial in R_p closest to $(p/q) \cdot c$ such that $c' \equiv c \pmod{2}$. Then, for any f with $\|[fc]_q\|_\infty < q/2 - (q/p) \cdot \|f\|_1$, we have*

$$[fc']_p = [fc]_q \pmod{2} \quad \text{and} \quad \|[fc']_p\|_\infty < (p/q) \cdot \|[fc]_q\|_\infty + \|f\|_1$$

where $\|\cdot\|_\infty$ and $\|\cdot\|_1$ are the ℓ_∞ and ℓ_1 , respectively.

Proof. Let $fc = \sum_{i=0}^{n-1} d_i x^i$, and consider a coefficient d_i . We know that there exists $k \in \mathbb{Z}$ such that:

$$[d_i]_q = d_i - kq \in \left[-\frac{q}{2} + \frac{q}{p} \|f\|_1, \frac{q}{2} - \frac{q}{p} \|f\|_1 \right],$$

so that

$$(p/q) \cdot d_i - kp \in \left[-\frac{p}{2} + \|f\|_1, \frac{p}{2} - \|f\|_1 \right]$$

Let $fc' = \sum_{i=0}^{n-1} e_i x^i$. Then $-\|f\|_1 \leq (p/q) \cdot e_i - d_i \leq \|f\|_1$. Therefore,

$$e_i - kp \in \left[-\frac{p}{2}, \frac{p}{2} \right] \quad \text{and} \quad [e_i]_p = e_i - kp$$

This proves the second part of the lemma. To prove the first part, notice that since p and q are both odd, we know $kp \equiv kq \pmod{2}$. Moreover, we chose c' such that $c \equiv c' \pmod{2}$. We thus have

$$\begin{aligned} e_i - kp &\equiv d_i - kq \pmod{2} \\ [e_i]_p &\equiv [d_i]_q \pmod{2} \\ [fc']_p &\equiv [fc]_q \pmod{2} \end{aligned}$$

□

The beauty of Lemma 3.5 is that if we know the depth D of the circuit we want to evaluate, then we can construct a ladder of decreasing moduli q_0, \dots, q_D and perform modulus reduction after each operation so that at level i all ciphertexts reside in R_{q_i} and the magnitude of the noise at each level is small. In particular, this is true at level D so that once the evaluation is complete, it is possible to decrypt the resulting ciphertext without decryption errors. This yields a *leveled homomorphic encryption scheme*. A bootstrappable scheme can then be obtained by setting $D = D_{\text{dec}}$, the depth of the augmented decryption circuit.

3.4.2 Obtaining A Leveled Homomorphic Scheme

We change the somewhat homomorphic scheme from Section 3.3 to use modulus reduction during the evaluation. The main changes to the scheme are as follows:

- The scheme is now additionally parametrized by an integer D , which is the maximum circuit depth that it can homomorphically evaluate, and a ladder of decreasing moduli q_0, \dots, q_D .
- We cannot use a single key f for all levels (at the expense of assuming the circular security), as in Section 3.3. This is because the public key h depends on the modulus q (recall that $h = 2gf^{-1}$, where f^{-1} is the inverse of f in R_q). With the new ladder of moduli, this

would require that the following two conditions be met simultaneously: First, that f^{-1} is the inverse of f in R_{q_D} (to guarantee correctness of decryption) and second, that $h = 2gf^{-1}$ is (indistinguishable from) uniformly random in R_{q_0} (to guarantee semantic security). This would require making a much stronger (and perhaps false) hardness assumption.

Instead, key generation computes a different key pair $(h^{(d)}, f^{(d)})$ for each level $d \in \{0, \dots, D\}$. Encryption uses $\text{pk} \stackrel{\text{def}}{=} h^{(0)}$ as the public key, and decryption uses $\text{sk}^{(d)} \stackrel{\text{def}}{=} f^{(d)}$ to decrypt a “level- d ” ciphertext, ie. a ciphertext that is the output of a depth- d circuit evaluation. W.l.o.g. we assume any ciphertext to be decrypted is a level- D ciphertext and set the secret key to be $\text{sk} = f^{(D)}$.

Homomorphic operations will ensure that if c, c' are level- $(d-1)$ ciphertexts in $R_{q_{d-1}}$ decryptable with $f^{(d-1)}$, then c_{add} and c_{mult} are level- d ciphertexts in R_{q_d} decryptable with $f^{(d)}$.

- Relinearization will now serve two purposes: it will ensure that only linear terms of keys are needed to decrypt the resulting ciphertext, but it will also switch the level- $(d-1)$ key to the corresponding level- (d) key. (Indeed, relinearization is also known as *key-switching* in the literature). Moreover, note that we must perform the key-switching step not only for quadratic terms but also for linear terms. Thus, we now perform relinearization/key-switching after *every* homomorphic operation, both addition and multiplication, and furthermore, we relinearize/key-switch every key in $K \cup K'$, instead of only those in $K \cap K'$.
- To perform the relinearization/key-switching step described above, the evaluation key consists of pseudo-encryptions of $f^{(d-1)}$ and $(f^{(d-1)})^2$ under the public key $h^{(d)}$, for all $d \in [D]$.

Note in particular that we now need pseudo-encryptions of the quadratic terms of the key. In the scheme from Section 3.3, relinearization only required pseudo-encryptions of (multiples of) f because the term $\langle \text{Bit}(c), \text{Pow}(f) \rangle$ only performed “partial decryption” of the ciphertext c ; it computes fc but f^2 is required to decrypt c . Decryption of c was completed at decryption time when the ciphertext was multiplied by f once more, obtaining f^2c .

In our new setting, because decryption is performed using a *different* key, relinearization needs to “completely decrypt” c with the original key. For a key in $K \cap K'$, this means computing $\left[\left\langle \text{Bit}(c), \text{Pow}\left((f^{(d-1)})^2\right) \right\rangle \right]_q = \left[(f^{(d-1)})^2 c \right]_q$. Since $\text{Pow}\left((f^{(d-1)})^2\right)$ is encrypted under $h^{(d)}$, the new ciphertext can be decrypted using $f^{(d)}$.

Pseudo-encryptions of the linear terms of the keys are also required in order to relinearize/key-switch keys in $K \triangle K'$, the symmetric difference of K, K' .

3.4.3 Formal Description

We now give a formal description of the leveled homomorphic encryption scheme resulting from applying the changes described above to the somewhat homomorphic scheme \mathcal{E}_{SH} described in Section 3.3.

- **LH.Keygen** (1^κ) : For every $i \in \{0, \dots, D\}$, sample $g^{(i)}, u^{(i)} \leftarrow \chi$ and set $f^{(i)} := 2u^{(i)} + 1$ so that $f^{(i)} \equiv 1 \pmod{2}$. If $f^{(i)}$ is not invertible in R_{q_i} , resample $u^{(i)}$; otherwise, let $(f^{(i)})^{-1}$

be the inverse of $f^{(i)}$ in R_q . Let $h^{(i)} \stackrel{\text{def}}{=} \left[2g^{(i)} (f^{(i)})^{-1} \right]_{q_i} \in R_{q_i}$, and set

$$\text{pk} \stackrel{\text{def}}{=} h^{(0)} \in R_{q_0} \quad , \quad \text{sk} \stackrel{\text{def}}{=} f^{(D)} \in R_{q_D}$$

For all $i \in [D]$, sample $\tilde{\mathbf{s}}_\gamma^{(i)}, \tilde{\mathbf{e}}_\gamma^{(i)}, \tilde{\mathbf{s}}_\zeta^{(i)}, \tilde{\mathbf{e}}_\zeta^{(i)} \leftarrow \chi^{\lceil \log q \rceil}$ and compute

$$\begin{aligned} \gamma^{(i)} &:= \left[h^{(i)} \tilde{\mathbf{s}}_\gamma^{(i)} + 2\tilde{\mathbf{e}}_\gamma^{(i)} + \text{Pow} \left(f^{(i-1)} \right) \right]_{q_i} \in R_{q_i}^{\lceil \log q_i \rceil} \\ \zeta^{(i)} &:= \left[h^{(i)} \tilde{\mathbf{s}}_\zeta^{(i)} + 2\tilde{\mathbf{e}}_\zeta^{(i)} + \text{Pow} \left(\left(f^{(i-1)} \right)^2 \right) \right]_{q_i} \in R_{q_i}^{\lceil \log q_i \rceil} \end{aligned}$$

Set $\text{ek} \stackrel{\text{def}}{=} \left\{ \gamma^{(i)}, \zeta^{(i)} \right\}_{i \in [D]}$, and output the key tuple $(\text{pk}, \text{sk}, \text{ek})$.

- **LH.Enc**(pk, m) : Sample $s, e \leftarrow \chi$. Output the ciphertext $c := [hs + 2e + m]_{q_0} \in R_{q_0}$.
- **LH.Dec**($\text{sk}_1, \dots, \text{sk}_N, c$) : Assume w.l.o.g. that $c \in R_{q_D}$. Parse $\text{sk}_i = f_i$ for $i \in [N]$. Let $\mu := [f_1 \cdots f_N \cdot c]_{q_D} \in R_{q_D}$. Output $m' := \mu \pmod{2}$.
- **LH.Eval**($C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_\ell, \text{pk}_\ell, \text{ek}_\ell)$) : We show how to evaluate an ℓ -variate boolean circuit $C : \{0, 1\}^\ell \rightarrow \{0, 1\}$ of depth D . To this end, we show how to homomorphically add and multiply two elements in $\{0, 1\}$. As before, given two ciphertexts c, c' , we assume that we also have a set of distinct public keys associated with each ciphertext, and denote these lists by K, K' , respectively. The public-key set of a fresh encryption is simply the set $\{\text{pk}\}$ containing the public key under which it was encrypted. For convenience, in our analysis we sometimes assume that the set contains the indices of the public keys instead of the keys themselves.

- Given two ciphertexts $c, c' \in R_{q_d}$ with corresponding public-key sets K, K' , compute $c_0 = [c + c']_{q_d} \in R_{q_d}$ and let $K \cup K' = \{\text{pk}_{i_1}, \dots, \text{pk}_{i_t}\}$. For $j = 1, \dots, r$, parse $\text{ek}_{i_j} = \left\{ \gamma_{i_j}^{(\delta)}, \zeta_{i_j}^{(\delta)} \right\}_{\delta \in [D]}$ and compute

$$c_j = \left[\left\langle \text{Bit}(c_{j-1}), \gamma_{i_j}^{(d)} \right\rangle \right]_q \in R_{q_d}$$

Finally, reduce the modulus: let c_{add} be the integer vector closest to $(q_{d+1}/q_d) \cdot c_t$ such that $c_{\text{add}} \equiv c_t \pmod{2}$. Output $c_{\text{add}} \in R_{q_{d+1}}$ as an encryption of the *sum* of the underlying messages. Output the set $K_{\text{add}} \stackrel{\text{def}}{=} K \cup K'$ as its corresponding public-key set.

- Given two ciphertexts $c, c' \in R_{q_d}$ with corresponding public-key sets K, K' , compute $c_0 = [c + c']_{q_d} \in R_{q_d}$ and let $K \cup K' = \{\text{pk}_{i_1}, \dots, \text{pk}_{i_t}\}$. For $j = 1, \dots, r$, parse $\text{ek}_{i_j} = \left\{ \gamma_{i_j}^{(\delta)}, \zeta_{i_j}^{(\delta)} \right\}_{\delta \in [D]}$ and compute c_j as follows:
 - * If $\text{pk}_{i_j} \in K \cap K'$, let

$$c_j = \left[\left\langle \text{Bit}(c_{j-1}), \gamma_{i_j}^{(d)} \right\rangle \right]_q \in R_{q_d}$$

* Otherwise, let

$$c_j = \left[\left\langle \text{Bit}(c_{j-1}), \zeta_{i_j}^{(d)} \right\rangle \right]_q \in R_{q_d}$$

Finally, reduce the modulus: let c_{mult} be the integer vector closest to $(q_{d+1}/q_d) \cdot c_t$ such that $c_{\text{mult}} \equiv c_t \pmod{2}$. Output $c_{\text{mult}} \in R_{q_{d+1}}$ as an encryption of the *product* of the underlying messages. Output the set $K_{\text{mult}} \stackrel{\text{def}}{=} K \cup K'$ as its corresponding public-key set.

Leveled Homomorphism. We can now show the following lemma, characterizing the circuits and number of keys that the scheme can handle in evaluation.

Lemma 3.6. *Let χ is a B -bounded distribution for $B = \text{poly}(n)$, let $q_0 = 2^{n^\varepsilon}$ for $\varepsilon \in (0, 1)$ and for $d \in [D]$, let $q_{d-1}/q_d = 8n(nB)^{2N+2}$. Then the encryption scheme $\mathcal{E}_{\text{LH}} = (\text{LH.Keygen}, \text{LH.Enc}, \text{LH.Dec}, \text{LH.Eval})$ described above is multikey homomorphic for N keys and circuits of depth D as long as $ND = O(n^\varepsilon / \log n)$.*

Proof. We claim that for all $d \in \{0, \dots, D\}$, the error of a level- d ciphertext is bounded by $E \stackrel{\text{def}}{=} (1/2n) \cdot (q_{d-1}/q_d) = 4(nB)^{2N+2}$, and prove it by induction. The base case follows immediately since the error of a freshly encrypted ciphertext is bounded by $3(nB)^2 < 4(nB)^{2N+2}$.

We now turn to the inductive step. Let c, c' be level- $(d-1)$ ciphertexts with corresponding public key sets K, K' . The inductive hypothesis tells us the error in c and c' is bounded by E . Using the same analysis as in the proof of Lemma 3.2, we can show that relinearizing all keys in $K \cup K'$ generates an additive error less than $8 \lceil \log q_d \rceil (nB)^{2N+1} < (nB)^{2N+2}$, where we used the fact that $q_d < q_0 = 2^{n^\varepsilon}$ for $\varepsilon < 1$. Recall that c_t is the ciphertext obtained in a homomorphic operation after relinearization has been completed but before modulus reduction is performed. Then:

- In a homomorphic addition, the error of c_t is bounded by $2(nB)^N E + (nB)^{2N+2}$. By Lemma 3.5, the error of c_{add} is bounded by:

$$\begin{aligned} \frac{q_d}{q_{d-1}} \cdot (2(nB)^N E + (nB)^{2N+2}) + \|f\|_1 &\leq \frac{2(nB)^N E + (nB)^{2N+2}}{2nE} + nB \\ &\leq \frac{2(nB)^N E}{2nE} + (nB)^{2N+2} + nB \\ &\leq \frac{(nB)^N}{n} + (nB)^{2N+2} + nB \\ &\leq 4(nB)^{2N+2} = E \end{aligned}$$

- In a homomorphic multiplication, the error of c_t is bounded by $nE^2 + (nB)^{2N+2}$. By Lemma 3.5, the error of c_{mult} is bounded by:

$$\begin{aligned} \frac{q_d}{q_{d-1}} \cdot (nE^2 + (nB)^{2N+2}) + \|f\|_1 &\leq \frac{nE^2 + (nB)^{2N+2}}{2nE} + nB \\ &\leq \frac{nE^2}{2nE} + 2(nB)^{2N+2} \\ &\leq \frac{E}{2} + \frac{E}{2} = E \end{aligned}$$

This concludes the inductive step and the proof that ciphertexts of all levels have error at most E .

To correctly decrypt a level- D ciphertext, we must have that

$$(nB)^{2N+2} = E < \frac{q_D}{2} < \frac{q_0}{2(8n(nB)^{2N+2})^D} = \frac{2^{n^\varepsilon}}{2(8n(nB)^{2N+2})^D}$$

which yields the theorem statement: $ND = O(n^\varepsilon / \log n)$. \square

Security. As in Section 3.3, the security of the scheme \mathcal{E}_{LH} can be based in the $\text{DSPR}_{\phi,q,\chi}$ and $\text{RLWE}_{\phi,q,\chi}$ assumptions. However, unlike in Section 3.3, we do not need to assume circular security of the encryption scheme. This is due to the fact that the evaluation key consists of pseudo-encryptions of (multiples of) $f^{(d-1)}$ and $(f^{(d-1)})^2$ under a *different* public key $h^{(d)}$, for all $d \in [D]$. Semantic security even given the evaluation key can then be established by a hybrid argument that converts all pseudo-encryptions in the evaluation key, one-by-one, to uniform elements in R_q .

Lemma 3.7. *Let n be a power of 2, let $\phi(x) = x^n + 1$, let $q = 2^{n^\varepsilon}$ for $\varepsilon \in (0, 1)$ and $\chi = D_{\mathbb{Z}^n, r}$ with $r = \text{poly}(n)$. Then the multikey leveled homomorphic encryption scheme $\mathcal{E}_{\text{LH}} = (\text{LH.Keygen}, \text{LH.Enc}, \text{LH.Dec}, \text{LH.Eval})$ described above is secure under the $\text{DSPR}_{\phi,q,\chi}$ and $\text{RLWE}_{\phi,q,\chi}$ assumptions.*

3.4.4 Multikey Fully Homomorphic Encryption

To convert the leveled homomorphic encryption scheme described in Section 3.4.2 into a fully homomorphic scheme, we use the multikey bootstrapping theorem (Theorem 3.4). First, we show an upper bound on the depth of the decryption circuit and show that the scheme is bootstrappable.

Lemma 3.8. *The N -key decryption circuit of the leveled homomorphic encryption scheme described above can be implemented as a polynomial-size arithmetic circuit over $GF(2)$ of depth $O(\log N \cdot (\log \log q_D + \log n))$.*

Proof. The decryption circuit for a ciphertext encrypted under N keys can be written as

$$\text{Dec}(f_1, \dots, f_N, c) = c \cdot \prod_{i=1}^N f_i$$

Multiplying two polynomials over R_{q_D} can be done using a polynomial-size Boolean circuit of depth $O(\log \log q_D + \log n)$ (see, e.g., [BV11a, Lemma 4.5] for a proof). Using a binary tree of polynomial multiplications, the decryption operation above can then be done in depth $O(\log N \cdot (\log \log q_D + \log n))$, as claimed. \square

This means that the modified scheme is bootstrappable, and therefore applying the bootstrapping theorem gives us:

Theorem 3.9. *Let χ is a B -bounded distribution for $B = \text{poly}(n)$, let $q_0 = 2^{n^\varepsilon}$ for $\varepsilon \in (0, 1)$ and for $d \in [D]$, let $q_{d-1}/q_d = 8n(nB)^{2N+2}$. Then, there exists a multikey fully homomorphic encryption scheme for $N = O(n^\varepsilon / \log^3 n)$ keys, secure under the $\text{DSPR}_{\phi,q,\chi}$ and $\text{RLWE}_{\phi,q,\chi}$ assumptions, and the assumption that the leveled homomorphic encryption scheme $\mathcal{E}_{\text{LH}} = (\text{LH.Keygen}, \text{LH.Enc}, \text{LH.Dec}, \text{LH.Eval})$ described above is weakly circular secure.*

Proof. To apply the multikey bootstrapping theorem (Theorem 3.4), we require that the depth of the decryption circuit is smaller than the depth of the circuits that the scheme can evaluate. That is, we require that

$$\log N \cdot (\log \log q_D + \log n) < C \cdot \frac{\log q_0}{N \cdot \log n}$$

for some universal constant $C > 0$. For $N \leq \sqrt{C/2} \cdot (n^{\varepsilon/2}/\log n)$, we have,

$$\begin{aligned} N \cdot \log n \cdot \log N \cdot (\log \log q_D + \log n) &\leq N^2 \cdot \log n \cdot (\log \log q_0 + \log n) \\ &\leq \frac{C}{2} \cdot \frac{n^\varepsilon}{\log^2 n} \cdot (1 + \varepsilon) \cdot \log^2 n \\ &\leq C \cdot n^\varepsilon = C \cdot \log q_0 \end{aligned}$$

as required. \square

Remark 3.4. *Theorem 3.9 implies that for any $N \in \mathbb{N}$, there exists a multikey fully homomorphic encryption scheme for N keys. This is achieved by choosing ε' such that $n^{\varepsilon'} \leq \sqrt{C/2} \cdot (n^{\varepsilon/2}/\log n)$ and setting $n \geq N^{1/\varepsilon'}$.*

We emphasize the fact that bootstrapping (and therefore assuming weak circular security) can be avoided at the cost of obtaining a leveled homomorphic encryption scheme.

4 On-the-Fly MPC from Multikey FHE

We now show how to construct on-the-fly MPC from multikey FHE. We first construct a basic protocol that is secure against semi-malicious adversaries, and then describe how to modify the protocol to obtain security against malicious adversaries. As mentioned earlier, the main building block of our construction is multikey fully homomorphic encryption, defined and constructed in Section 3.

4.1 The Basic Protocol

Let $\{\mathcal{E}^{(N)} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})\}_{N>0}$ be a multikey fully-homomorphic family of encryption schemes. The following construction is an on-the-fly MPC protocol secure against semi-malicious adversaries. The protocol is defined as follows:

Step 1: For $i \in [U]$, party P_i samples a key tuple $(\text{pk}_i, \text{sk}_i, \text{ek}_i) \leftarrow \text{Keygen}(1^\kappa)$ and encrypts its input x_i under pk_i : $c_i \leftarrow \text{Enc}(\text{pk}_i, x_i)$. It sends $(\text{pk}_i, \text{ek}_i, c_i)$ to the server S .

At this point a function F , represented as a circuit C , has been selected on inputs $\{x_i\}_{i \in V}$ for some $V \subseteq U$. Let $N = |V|$. For ease of notation, assume w.l.o.g. that $V = [N]$. The parties proceed as follows.

Step 2: The server S computes $c := \text{Eval}(C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_N, \text{pk}_N, \text{ek}_N))$ and broadcasts c to parties P_1, \dots, P_N .

Step 3: The parties P_1, \dots, P_N run a secure MPC protocol $\Pi_{\text{DEC}}^{\text{SM}}$ to compute the function $g_c(\text{sk}_1, \dots, \text{sk}_N) \stackrel{\text{def}}{=} \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c)$.

We remark that an upper bound on the number of computing parties must be known in advance. This is a direct consequence of the “leveled” nature of our multikey FHE construction with respect to the number of keys.

4.1.1 Security Against Semi-Malicious Adversaries

Theorem 4.1. *Let $\{\mathcal{E}^{(N)} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})\}_{N \geq 0}$ be a multikey fully-homomorphic encryption scheme, and let $\Pi_{\text{DEC}}^{\text{SM}}$ be an N -party MPC protocol for computing the decryption function $g_c(\text{sk}_1, \dots, \text{sk}_N) \stackrel{\text{def}}{=} \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c)$. If \mathcal{E} is semantically secure, and $\Pi_{\text{DEC}}^{\text{SM}}$ is secure against semi-honest adversaries corrupting $t < N$ parties, then the above construction is an on-the-fly MPC protocol secure against (static) semi-malicious adversaries corrupting t parties and possibly the server S .*

Proof. We prove that the protocol is correct and secure, and that it satisfies the performance requirements of an *on-the-fly* protocol.

Correctness: Correctness follows directly from the correctness properties of homomorphic evaluation and the MPC protocol $\Pi_{\text{DEC}}^{\text{SM}}$ for decryption.

Performance: By compactness of evaluation, we know that c is independent of $|C|$. This means that the communication complexity and the computation time of the parties is independent of $|C|$.

Security: We show security for the case when the server is corrupted; the case when the server is honest is analogous. Let \mathcal{A}^{SM} be a real-world semi-malicious adversary corrupting t clients and the server. Recall that for security, we only need to consider adversaries corrupting a subset T of the parties P_1, \dots, P_N involved in the computation. Thus, we assume $t < N$, let $T \subsetneq [N]$ be the set of corrupted clients, and let $\bar{T} = [N] \setminus T$.

We construct a simulator \mathcal{S}^{SM} as follows. The simulator receives the inputs of the corrupted parties, $\{x_i\}_{i \in T}$ and runs \mathcal{A}^{SM} on these inputs $\{x_i\}_{i \in T}$. It simulates the messages for all honest parties in the protocol execution with \mathcal{A}^{SM} by sampling all key tuples correctly, but encrypting 0 instead of the honest input x_i (which it doesn’t know). In Step 3, it runs the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ for the protocol $\Pi_{\text{DEC}}^{\text{SM}}$.

Step 1: For non-computing parties $i \in \{N+1, \dots, U\}$ and for honest parties $i \in \bar{T}$, \mathcal{S}^{SM} computes $(\text{pk}_i, \cdot, \text{ek}_i) \leftarrow \text{Keygen}(1^\kappa)$ honestly and computes $c_i \leftarrow \text{Enc}(\text{pk}_i, 0)$. For each party P_i , the simulator sends $(c_i, \text{pk}_i, \text{ek}_i)$ to \mathcal{A}^{SM} on behalf of P_i .

At the end of this round, it reads from \mathcal{A}^{SM} ’s witness tape the secret keys $\{\text{sk}_i\}_{i \in T}$ and the inputs $\{\tilde{x}_i\}_{i \in T}$. The simulator sends these inputs to the trusted functionality \mathcal{F} and receives the output $\tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_N)$, where $\tilde{x}_i = x_i$ for honest inputs $i \in \bar{T}$.

Step 2: The simulator receives c from \mathcal{A}^{SM} as the server’s broadcast message.

Step 3: The simulator \mathcal{S}^{SM} runs the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ for the decryption protocol (interacting with \mathcal{A}^{SM}). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ queries the ideal decryption functionality with secret keys $\{\tilde{\text{sk}}_i\}_{i \in T}$, \mathcal{S}^{SM} returns \tilde{y} .

Output: The simulator receives the output of the corrupted parties from \mathcal{A}^{SM} , and returns these as its output.

We prove that $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{SM}}}(\vec{x}) \stackrel{c}{\approx} \text{REAL}_{\Pi^{\text{SM}}, \mathcal{A}^{\text{SM}}}(\vec{x})$ via a series of hybrids.

Hybrid 0: This is the real-world execution of the protocol.

Hybrid 1: We change how Step 3 is performed. Instead of executing the protocol $\Pi_{\text{DEC}}^{\text{SM}}$ where honest parties use their individual secret keys, we run the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ (interacting with \mathcal{A}^{SM}). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ queries the ideal decryption functionality with secret keys $\{\tilde{\text{sk}}_i\}_{i \in T}$, we return

$$\tilde{y} = g_c(\tilde{\text{sk}}_1, \dots, \tilde{\text{sk}}_N) = \text{Dec}(\tilde{\text{sk}}_1, \dots, \tilde{\text{sk}}_N, c)$$

where $\tilde{\text{sk}}_i = \text{sk}_i$ for honest secret keys $i \in \bar{T}$. The output of the corrupted parties is defined to be the output of $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$, and the output of the honest parties is defined to be \tilde{y} .

We claim that Hybrid 0 is computationally indistinguishable from Hybrid 1 by the *security of $\Pi_{\text{DEC}}^{\text{SM}}$* . Indeed, the security of the decryption protocol $\Pi_{\text{DEC}}^{\text{SM}}$ guarantees that as long as we correctly emulate the ideal decryption functionality, the joint output of all parties is computationally indistinguishable in a real-world execution of the protocol with adversary \mathcal{A}^{SM} (Hybrid 0), and in an ideal-world execution of the protocol with adversary $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ (Hybrid 1). We correctly emulate the ideal decryption functionality, by definition.

Hybrid 2: We now change how we compute \tilde{y} , the value returned to the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ when it queries the decryption ideal functionality. Instead of computing $\tilde{y} = g_c(\tilde{\text{sk}}_1, \dots, \tilde{\text{sk}}_N) = \text{Dec}(\tilde{\text{sk}}_1, \dots, \tilde{\text{sk}}_N, c)$, we instead compute

$$\tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_N)$$

where $\tilde{x}_i = x_i$ for honest inputs $i \in \bar{T}$, and where for corrupt parties $i \in T$, we recover \tilde{x}_i by reading \mathcal{A}^{SM} 's witness tape at the end of Step 1.

We claim that Hybrid 1 and Hybrid 2 are identically distributed. The adversary \mathcal{A}^{SM} follows the protocol as specified, so in particular, it performs the homomorphic evaluation correctly. By *correctness of multikey evaluation* we know that c decrypts to $f(\tilde{x}_1, \dots, \tilde{x}_N)$ when decrypted using the secret keys it computed in Step 1, $\{\text{sk}_i\}_{i \in [N]}$; that is, $\text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c) = f(\tilde{x}_1, \dots, \tilde{x}_N)$.

Furthermore, because the adversary \mathcal{A}^{SM} follows the protocol as specified, we know that the secret keys it uses in Step 3 are the same as the ones it computed in Step 1, i.e. $\text{sk}_i = \tilde{\text{sk}}_i$ for all $i \in T$. We conclude that $\text{Dec}(\tilde{\text{sk}}_1, \dots, \tilde{\text{sk}}_N, c) = f(\tilde{x}_1, \dots, \tilde{x}_N)$.

Hybrids 3.k for $k = 1, \dots, N - t$: Let $\bar{T} = \{i_1, \dots, i_{N-t}\}$. In Hybrid 3.k we change c_{i_k} so that instead of encrypting x_{i_k} it now encrypts 0. More formally, in Hybrid 3.k we have:

$$\left\{ c_{i_j} \leftarrow \text{Enc}(\text{pk}_{i_j}, 0) \right\}_{j \leq k}, \quad \left\{ c_{i_j} \leftarrow \text{Enc}(\text{pk}_{i_j}, x_{i_j}) \right\}_{j > k}$$

For ease of notation we let Hybrid 2 be Hybrid 3.0. We claim that the view of \mathcal{A}^{SM} in Hybrid 3. k is indistinguishable from its view in Hybrid 3. $(k-1)$ by the *semantic security of \mathcal{E}* under public key pk_{i_k} . Indeed, now that we run the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ in Step 3 instead of the real decryption protocol, the secret key sk_{i_k} is only used to encrypt c_{i_k} . So suppose, for the sake of contradiction, that there exists an algorithm \mathcal{D} that distinguishes between hybrids 3. k and 3. $(k-1)$. We construct an adversary \mathcal{B} that breaks the semantic security of \mathcal{E} under public key pk_{i_k} . The reduction \mathcal{B} works as follows:

1. The reduction chooses arbitrary $\{x_i\}$.
2. It receives (pk, ek) from the semantic security challenger and sets $\text{pk}_{i_k} = \text{pk}$ and $\text{ek}_{i_k} = \text{ek}$. Gives $m_0 = 0$ and $m_1 = x_{i_k}$ to the challenger and receives $c = \text{Enc}(\text{pk}, m_b)$. Sets $c_{i_k} = c$. For all $i \in \bar{T}, i \neq i_k$, computes $(\text{pk}_i, \cdot, \text{ek}_i) \leftarrow \text{Keygen}(1^\kappa)$ honestly. For $j < k$, computes $c_{i_j} \leftarrow \text{Enc}(\text{pk}_{i_j}, 0)$ and for $j > k$, computes $c_{i_j} \leftarrow \text{Enc}(\text{pk}_{i_j}, x_{i_j})$.
3. The reduction runs \mathcal{A}^{SM} : for all $i \in \bar{T}$ gives $(\text{pk}_i, \text{ek}_i, c_i)$ to \mathcal{A}^{SM} on behalf of P_i , and receives c from \mathcal{A}^{SM} .
4. It reads from \mathcal{A}^{SM} 's witness tape the inputs $\{\tilde{x}_i\}_{i \in T}$ and runs the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ (interacting with \mathcal{A}^{SM}). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$ queries the ideal decryption functionality, it returns $\tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_N)$ where $\tilde{x}_i = x_i$ for inputs $i \in \bar{T}$.
5. The reduction then gives $\mathcal{D} \tilde{y}$ as the output of all honest parties, as well as the output of $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{SM}}$.
6. Finally, \mathcal{B} outputs the bit output by \mathcal{D} .

When $b = 0$, \mathcal{B} perfectly emulates Hybrid 3. k , whereas if $b = 1$, \mathcal{B} perfectly emulates Hybrid 3. $(k-1)$. Therefore, if \mathcal{D} can distinguish between Hybrids 3. k and 3. $(k-1)$, then \mathcal{B} can distinguish between an encryption of m_0 and an encryption of m_1 , contradicting the semantic security of \mathcal{E} .

We have proved that the joint output in Hybrid 0 is computationally indistinguishable from the joint output in Hybrid 3. $(N-t)$. But notice that the joint output in Hybrid 3. $(N-t)$ is precisely $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{SM}}}(\vec{x})$, and the joint output in Hybrid 0 is defined to be $\text{REAL}_{\Pi^{\text{SM}}, \mathcal{A}^{\text{SM}}}(\vec{x})$. We conclude that $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{SM}}}(\vec{x}) \stackrel{c}{\approx} \text{REAL}_{\Pi^{\text{SM}}, \mathcal{A}^{\text{SM}}}(\vec{x})$, as desired. □

4.2 Achieving Security Against Malicious Adversaries

The protocol described in Section 4.1, though secure against semi-malicious adversaries, is not secure against fully malicious adversaries. We transform the protocol into one that is secure against malicious corruptions in three steps:

1. First, we replace the decryption protocol in Step 3 with one that is secure against malicious corruptions. More importantly, we change the function it computes to ensure that the secret key used in this protocol is consistent with the public and evaluation keys that the parties computed in Step 1.

2. Second, we add zero-knowledge proofs at each step in the protocol, following the AJW compiler [AJW11, AJL⁺12] (which is based on the GMW compiler [GMW87]).
3. Finally, in order to maintain the performance guarantees of the scheme, in Step 2 we replace the server's proof with a *succinct argument* (not necessarily ZK). This allows the server to prove that it correctly performed the homomorphic evaluation, and the clients to verify the validity of the proof in time that is significantly less than the size of the circuit.

The New Decryption Protocol. Our first step in handling malicious attacks is to replace the decryption protocol $\Pi_{\text{DEC}}^{\text{SM}}$ with one that is secure against malicious adversaries; we will denote it by $\Pi_{\text{DEC}}^{\text{MAL}}$. The function being computed by this protocol also needs to change in order to guarantee that the secret key used by each party is consistent with its public and evaluation keys:

$$g_{c, \text{pk}_1, \text{ek}_1, \dots, \text{pk}_N, \text{ek}_N}((\text{sk}_1, r_1) \dots, (\text{sk}_N, r_N)) \\ \stackrel{\text{def}}{=} \begin{cases} \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c) & \text{if } (\text{pk}_i, \text{sk}_i, \text{ek}_i) = \text{Keygen}(1^\kappa; r_i) \quad \forall i \in [N] \\ \perp & \text{otherwise} \end{cases}$$

Intuitively, if the protocol outputs something other than \perp , then in particular every corrupt party P_i “knows” a secret key $\tilde{\text{sk}}_i$ that is consistent with its public and evaluation keys $(\text{pk}_i, \text{ek}_i)$. By correctness of decryption, this binds P_i to the input $\tilde{x}_i = \text{Dec}(\tilde{\text{sk}}_i, \tilde{c}_i)$, which by semantic security of the FHE, must be independent of the honest party's inputs.

We remark that the proceedings version of this work [LTV12] does not change the decryption function, but instead adds to Step 1 a zero-knowledge proof π_i^{GEN} for the relation $R^{\text{GEN}} = \{ (\text{pk}_i, \text{ek}_i), (\text{sk}_i, r_i) \mid (\text{pk}_i, \text{sk}_i, \text{ek}_i) := \text{Keygen}(1^\kappa; r_i) \}$. While this guarantees that the public and evaluation keys are well-formed, it does not guarantee that the secret key used in the decryption protocol in Step 3 is consistent with the public and evaluation keys $(\text{pk}_i, \text{ek}_i)$ created and used in Step 1. This allows a corrupt party to use a different secret key sk_i^* in Step 3 and potentially change the outcome of the decryption. We are therefore unable to prove security of that construction. However, the zero-knowledge proofs π_i^{GEN} can be required as an *optimization*, to guarantee that an honest server does not accept, store, or compute on ciphertexts that are encrypted under malformed keys (even though the outcome of any joint computation on such a ciphertext would not be decryptable using protocol $\Pi_{\text{DEC}}^{\text{MAL}}$).

Finally, we highlight the fact that if the protocol $\Pi_{\text{DEC}}^{\text{MAL}}$ can be implemented using the cloud-assisted protocol of Asharov et al. [AJW11, AJL⁺12]. Jumping ahead, this yields a 5-round on-the-fly MPC protocol in the CRS-model, secure against malicious corruptions of any $t < [N]$ parties and possibly the server.

Adding Zero-Knowledge Proofs. The second step in our transformation is to apply the AJW compiler [AJW11, AJL⁺12] (based on the GMW compiler [GMW87]) to the rest of the protocol (Steps 1 and 2), in order to ensure that parties do not deviate from the protocol specifications. This entails having each party and the server compute a zero-knowledge proof at every round, proving that their message in that round is well-formed and consistent with the protocol transcript.

Because the well-formedness of the public and evaluation keys $(\text{pk}_i, \text{ek}_i)$ is checked in the decryption protocol $\Pi_{\text{DEC}}^{\text{MAL}}$, the parties do not need to compute a separate zero-knowledge proof for this statement (unless required for the optimization described above). Therefore, each party only

needs to prove that their ciphertext c_i is well-formed by providing a non-interactive zero-knowledge (NIZK) proof for the NP relation:

$$R^{\text{ENC}} = \{ ((\text{pk}_i, c_i) , (x_i, s_i)) \mid c_i = \text{Enc}(\text{pk}_i, x_i ; s_i) \}$$

We highlight the fact that the proof π_i^{ENC} must be *non-interactive*, for reasons that will become apparent shortly. Informally, this proof will either be broadcast by the server in Step 2 for all parties to verify, or it will be used as a witness in the proof of another NP relation. An interactive zero-knowledge proof would not be convincing in either of these cases since a valid proof transcript can always be simulated without knowing a witness and without the use of any trapdoors.

Maintaining Performance Guarantees. Unfortunately, verifying a standard zero-knowledge proof for the server’s computation in Step 2 requires time proportional to the size of the circuit. On the other hand, this computation is deterministic and public; indeed, anyone can verify the validity of the server’s broadcast message by performing the homomorphic evaluation themselves, albeit by also computing in time proportional to the size of the circuit. We solve this problem by replacing the server’s proof with a *succinct argument* (not necessarily ZK), that allows the server to prove that it correctly performed the homomorphic evaluation, and the clients to verify the validity of the proof in time that is significantly less than the size of the circuit. We offer several solutions, each with its own benefits and drawbacks.

Verification for Small Inputs: We first consider the case where the ciphertexts (c_1, \dots, c_N) are small enough to be broadcast to the N parties in V , allowing communication complexity in the online phase to be linear in the total input size of the participating parties. In this case, the server will broadcast all ciphertexts and proofs $\{c_i, \pi_i^{\text{ENC}}\}_{i \in [N]}$, the evaluated ciphertext c , and a succinct argument φ showing that it performed the homomorphic evaluation correctly. The server needs to convince the participating parties that “ $c = \text{Eval}(C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_N, \text{pk}_N, \text{ek}_N))$ ”, i.e., that a deterministic circuit of size $\text{poly}(|C|, \kappa)$ accepts. For any uniform circuit C (i.e., computable by a $\text{poly}(\kappa)$ -time Turing machine), the following offer $\text{poly}(\kappa, \log(|C|))$ communication and verification efficiency.¹¹

1. Use the argument system of Kilian [Kil92, Kil95], yielding *interactive* 4-round verification. It relies on expensive PCPs.
2. Use the succinct non-interactive arguments (SNARGs and SNARKs) of Micali [Mic94], Bitansky et al. [BCCT12, BCCT13] or Goldwasser et al. [GLR11] (see Section 2.3). These are non-interactive¹² but are secure only in the random oracle model [BR93] (in the case of CS proofs) or hold in the standard model but require a non-falsifiable assumption [Nao03]. Some variants rely on PCPs, PIR or FHE.

In case that the evaluation circuit is in logspace-uniform **NC**, we have another alternative:

¹¹For any given family of C , $|C| = \text{poly}(\kappa)$, and thus, $\text{poly}(\kappa, \log(|C|)) = \text{poly}(\kappa)$; but the degree of this polynomial depends on the circuit family.

¹²In our protocol, each party can run **Gen** in Step 1 and send the **vrs** to the server in that step. Or in the case of CS proofs, where only a description of a hash function is required, this can be added to the CRS of the protocol.

4. Use the argument system of Goldwasser et al. [GKR08] for a 1-round solution¹³. It relies on PIR.

Unfortunately, we are unable to use verifiable computation protocols in the pre-processing model (e.g. [GGP10, CKV10, AIK10]) or SNARGs/SNARKs where the CRS depends on the circuit to be computed or where its size is at least as big as the computation, e.g. [Gro10, Lip12, GGPR13, PHGR13, Lip13]. These require the clients to participate in a pre-processing phase where their computation is proportional to the size of the circuit, violating the performance requirements of on-the-fly MPC. Moreover, with this pre-processing step the model loses its dynamic nature, where users can compute many different functions on their inputs and can choose these functions dynamically, “on-the-fly”. Indeed, using these solutions would limit the parties to only compute functions for which they have already performed the corresponding pre-processing work or computed the corresponding CRS.

Verification for Large Inputs: We can make the communication and verification complexities depend merely polylogarithmically on the size of the relevant inputs x_1, \dots, x_N . This requires a succinct argument system that is a *proof of knowledge*. This is satisfied by Micali’s construction of CS proofs under Valiant’s analysis [Mic94, Val08], and by SNARKs [BCCT12, BCCT13]. The complexity of these arguments depends polynomially on the size of the statement being proven, but merely polylogarithmically on the size of the witness for the statement. We thus move c_i from the instance into the witness. To recognize the correct c_i , each party P_i remembers the digest of c_i under a collision-resistant hash function family $\mathcal{H} = \{H_{\text{hk}} : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa\}$.

In the offline stage, every party P_i samples a hash key hk_i and computes the digest $d_i = H_{\text{hk}_i}(c_i)$. Party P_i then sends $(c_i, \pi_i^{\text{ENC}}, \text{hk}_i, d_i)$ to the cloud. Each party P_i remembers its own (hk_i, d_i) pair but can forget the potentially long $x_i, c_i, \pi_i^{\text{ENC}}$. In the online stage, the server broadcasts $(\text{hk}_1, d_1), \dots, (\text{hk}_N, d_N)$ and proves the following NP statement: “there exist $\tilde{c}_1, \tilde{\pi}_1^{\text{ENC}}, \dots, \tilde{c}_N, \tilde{\pi}_N^{\text{ENC}}$ such that $d_i = H_{\text{hk}_i}(\tilde{c}_i)$ and $c = \text{Eval}(C, (\tilde{c}_1, \text{pk}_1, \text{ek}_1), \dots, (\tilde{c}_N, \text{pk}_N, \text{ek}_N))$ and $\tilde{\pi}_i^{\text{ENC}}$ is a valid proof”.

The construction is secure, since whenever the server convinces the clients, it actually “knows” such $\tilde{c}_1, \tilde{\pi}_1^{\text{ENC}}, \dots, \tilde{c}_N, \tilde{\pi}_N^{\text{ENC}}$ which can be efficiently extracted from the server (by the arguments’ proof of knowledge property). For an honest party, the extracted \tilde{c}_i must be the one originally sent by the party (by the collision-resistance of H). For a corrupt party, the extracted \tilde{c}_i must be a valid ciphertext (by the soundness of $\tilde{\pi}_i^{\text{ENC}}$) and its plaintext can be efficiently extracted using the secret key used by P_i in the decryption protocol in Step 3.

4.2.1 Formal Protocol

We now write a formal description of our construction of on-the-fly MPC, secure against malicious adversaries, and providing correct verification for large inputs. Our construction requires the following building blocks:

¹³The protocol has 2 rounds, but (as in the case of SNARGs and SNARKs) the first round is a challenge that is independent of the language and the statement, and can therefore be precomputed by the clients in Step 1 of our protocol. Each challenge can only be used for one proof, so the client must refresh the challenge after each computation.

- A semantically-secure multikey fully-homomorphic family of encryption schemes $\mathcal{E} = \{\mathcal{E}^{(N)} = (\text{Keygen}, \text{Enc}, \text{Dec}, \text{Eval})\}_{N>0}$.
- A family of collision-resistant hash functions $\mathcal{H} = \{H_{\text{hk}} : \{0, 1\}^* \rightarrow \{0, 1\}^\kappa\}_{\text{hk}}$.
- A NIZK argument system $\Pi^{\text{ENC}} = (\text{Setup}^{\text{ENC}}, \text{Prove}^{\text{ENC}}, \text{Verify}^{\text{ENC}}, \text{Sim}^{\text{ENC}})$ for the NP relation $R^{\text{ENC}} = \{ ((\text{pk}, c), (x, s)) \mid c = \text{Enc}(\text{pk}, x; s) \}$.
- An adaptively extractable SNARK system $\Phi = (\text{Setup}^\Phi, \text{Prove}^\Phi, \text{Verify}^\Phi, \text{Ext}^\Phi)$ for all of NP.
- An N -party MPC protocol, secure against malicious adversaries corrupting $t < N$ parties, for computing the family of decryption functions

$$g_{c, \text{pk}_1, \text{ek}_1, \dots, \text{pk}_N, \text{ek}_N}((\text{sk}_1, r_1) \dots, (\text{sk}_N, r_N)) \\ \stackrel{\text{def}}{=} \begin{cases} \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c) & \text{if } (\text{pk}_i, \text{sk}_i, \text{ek}_i) = \text{Keygen}(1^\kappa; r_i) \ \forall i \in [N] \\ \perp & \text{otherwise} \end{cases}$$

The protocol is defined as follows:

Input: All parties and the server receive as input the common reference string crs^{ENC} for the NIZK proof system Π^{ENC} . If CS proofs are used as the SNARK system, the (description) of the random-oracle hash function is also given to all parties and the server.

Step 1: For $i \in [U]$, party P_i samples a key tuple $(\text{pk}_i, \text{sk}_i, \text{ek}_i)$, encrypts its input x_i , and computes a NIZK showing that the ciphertext is well-formed:

$$(\text{pk}_i, \text{sk}_i, \text{ek}_i) := \text{Keygen}(1^\kappa; r_i) \quad , \quad c_i := \text{Enc}(\text{pk}_i, x_i; s_i) \\ \pi_i^{\text{ENC}} \leftarrow \text{Prove}^{\text{ENC}}((\text{pk}_i, c_i), (x_i, s_i))$$

It also samples a hash key hk_i and computes the digest of the ciphertext: $d_i = H_{\text{hk}_i}(c_i)$. It additionally creates a verification reference string and private verification key: $(\text{vrs}_i, \text{priv}_i) \leftarrow \text{Setup}^\Phi(1^\kappa)$.

Party P_i sends the tuple $(\text{pk}_i, \text{ek}_i, c_i, \pi_i^{\text{ENC}}, \text{hk}_i, d_i, \text{vrs}_i)$ to the server, who verifies all proofs $\{\pi_i^{\text{ENC}}\}_{i \in [U]}$.

From this point forward, party P_i can forget its (potentially long) input x_i , ciphertext c_i , and proof π_i^{ENC} . It need only remember its secret key and key-generation randomness (sk_i, r_i) , the hash key and digest (hk_i, d_i) , and its private verification key priv_i .

A function F , represented as a circuit C , is now selected on inputs $\{x_i\}_{i \in V}$ for some $V \subseteq U$. Let $N = |V|$. For ease of notation, we assume w.l.o.g. that $V = [N]$.

Step 2: The server S computes $c := \text{Eval}(C, (c_1, \text{pk}_1, \text{ek}_1), \dots, (c_N, \text{pk}_N, \text{ek}_N))$ and creates succinct arguments $\{\varphi_i\}_{i \in [N]}$ for the NP language

$$L = \{ \{(\text{pk}_i, \text{ek}_i, \text{hk}_i, d_i)\}_{i \in [N]} \mid \exists (\tilde{c}_1, \tilde{\pi}_1^{\text{ENC}}), \dots, (\tilde{c}_N, \tilde{\pi}_N^{\text{ENC}}) \text{ such that} \\ d_i = H_{\text{hk}_i}(\tilde{c}_i) \quad \text{and} \\ \text{Verify}^{\text{ENC}}((\text{pk}_i, \tilde{c}_i), \tilde{\pi}_i^{\text{ENC}}) = 1 \quad \text{and} \\ c = \text{Eval}(C, (\tilde{c}_1, \text{pk}_1, \text{ek}_1), \dots, (\tilde{c}_N, \text{pk}_N, \text{ek}_N)) \} \}$$

To compute φ_i , the server uses the verification reference string vrs_i . If CS proofs are used as the SNARK system, the server need only compute a single proof φ that can be verified by all.

The server broadcasts $(c, \varphi_1, \dots, \varphi_N)$ to all parties P_1, \dots, P_N , together with the tuple $\{(\text{pk}_i, \text{ek}_i, \text{hk}_i, d_i)\}_{i \in [N]}$.

Step 3: Party P_i runs $\text{Verify}^\Phi(\{(\text{pk}_i, \text{ek}_i, \text{hk}_i, d_i)\}_{i \in [N]}, \varphi_i)$ to verify the argument φ_i . If verification is successful for all parties, they run an MPC protocol $\Pi_{\text{DEC}}^{\text{MAL}}$ to compute the function

$$\begin{aligned} & g_{c, \text{pk}_1, \text{ek}_1, \dots, \text{pk}_N, \text{ek}_N}((\text{sk}_1, r_1) \dots, (\text{sk}_N, r_N)) \\ \stackrel{\text{def}}{=} & \begin{cases} \text{Dec}(\text{sk}_1, \dots, \text{sk}_N, c) & \text{if } (\text{pk}_i, \text{sk}_i, \text{ek}_i) = \text{Keygen}(1^\kappa; r_i) \ \forall i \in [N] \\ \perp & \text{otherwise} \end{cases} \end{aligned}$$

4.2.2 Proof of Security

Theorem 4.2. *Let $\mathcal{E}, \Pi_{\text{DEC}}^{\text{MAL}}, \mathcal{H}, \Pi^{\text{ENC}}, \Phi$ be as described in Section 4.2.1. Then the above construction is an on-the-fly MPC protocol secure against malicious adversaries corrupting $t < N$ parties and possibly the server S .*

Proof. We prove that the protocol is correct and secure, and that it satisfies the performance requirements of an *on-the-fly* protocol.

Correctness: Correctness follows directly from the correctness properties of homomorphic evaluation and the decryption MPC protocol $\Pi_{\text{DEC}}^{\text{MAL}}$.

Performance: The zero-knowledge proofs π_i^{ENC} are independent of C and the size of c is independent of $|C|$ by compactness of homomorphic evaluation. Moreover, the proof φ has size polylogarithmic in $|C|$ and its verification depends only polylogarithmically on the size of the ciphertexts c_i (and therefore polylogarithmically on the size of the inputs x_i as well). Thus, the communication complexity of the protocol is polylogarithmic in $|C|$, and the computation time of each party P_i is at most polylogarithmic in $|C|$ and the total size of the inputs, and polynomial in y and its input x_i .

Security: We show security for the case when the server is corrupted; the case when the server is honest is analogous. Let \mathcal{A}^{MAL} be a real-world semi-malicious adversary corrupting t clients and the server. Recall that for security, we only need to consider adversaries corrupting a subset T of the parties P_1, \dots, P_N involved in the computation. Thus, we assume $t < N$, let $T \subsetneq [N]$ be the set of corrupted clients, and let $\bar{T} = [N] \setminus T$.

We construct a simulator \mathcal{S}^{MAL} as follows. The simulator receives the inputs of the corrupted parties, $\{x_i\}_{i \in T}$ and runs \mathcal{A}^{MAL} on these inputs $\{x_i\}_{i \in T}$. It simulates the messages for all honest parties in the protocol execution with \mathcal{A}^{MAL} . In Step 1, it samples all key tuples correctly, but encrypts 0 instead of the honest input x_i (which it doesn't know), and computes simulated proofs π_i^{ENC} . In Step 2, it fixes an honest party h and extracts the witness $\{\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}}\}_{i \in [N]}$ of the argument φ_h . For all corrupted parties $i \in T$, the simulator extracts the corrupted input \tilde{x}_i from the proof $\tilde{\pi}_i^{\text{ENC}}$, submits these to the ideal functionality \mathcal{F} , and obtains an output \tilde{y} . In Step 3, it runs the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ for the protocol $\Pi_{\text{DEC}}^{\text{MAL}}$, returning \tilde{y} when it calls the ideal decryption functionality. More formally:

Step 1: The simulator creates the CRS for the NIZK Π^{ENC} , together with a trapdoor key and an extraction key:

$$(\text{crs}^{\text{ENC}}, \text{tk}^{\text{ENC}}, \text{extk}^{\text{ENC}}) \leftarrow \text{Setup}^{\text{ENC}}(1^\kappa)$$

For non-computing parties $i \in \{N+1, \dots, U\}$ and for honest parties $i \in \bar{T}$, the simulator computes $(\text{pk}_i, \cdot, \text{ek}_i) \leftarrow \text{Keygen}(1^\kappa)$ and samples hk_i honestly. The simulator also runs the verification setup honestly: $(\text{vrs}_i, \text{priv}_i) \leftarrow \text{Setup}^\Phi(1^\kappa)$.

The simulator computes an encryption of 0 and simulated zero-knowledge proofs:

$$c_i \leftarrow \text{Enc}(\text{pk}_i, 0) \quad , \quad \pi_i^{\text{ENC}} \leftarrow \text{Sim}(\text{tk}^{\text{ENC}}, (\text{pk}_i, c_i))$$

It computes the digest $d_i = H_{\text{hk}_i}(c_i)$ honestly. For each party P_i , \mathcal{S}^{MAL} sends $(\text{pk}_i, \text{ek}_i, c_i, \pi_i^{\text{ENC}}, \text{hk}_i, d_i, \text{vrs}_i)$ to \mathcal{A}^{MAL} on behalf of P_i .

Step 2: The simulator receives $(c, \varphi_1, \dots, \varphi_N)$ from \mathcal{A}^{MAL} , together with the tuples $\{(\text{pk}_i, \text{ek}_i, \text{hk}_i, d_i)\}_{i \in [N]}$. The simulator verifies φ_i for all honest parties $i \in \bar{T}$ and for a fixed honest party $h \in \bar{T}$, uses the SNARG extractor to extract witness $\{\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}}\}_{i \in [N]}$ from φ_h :

$$\{\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}}\}_{i \in [N]} \leftarrow \text{Ext}^\Phi \left(\{(\text{pk}_i, \text{ek}_i, \text{hk}_i, d_i)\}_{i \in [N]}, \varphi_h \right)$$

It outputs \perp if for any $i \in [N]$, verification fails for φ_i or $\tilde{\pi}_i^{\text{ENC}}$, or if $d_i \neq H_{\text{hk}_i}(\tilde{c}_i)$. It also outputs \perp if $c \neq \text{Eval}(C, (\tilde{c}_1, \text{pk}_1, \text{ek}_1), \dots, (\tilde{c}_N, \text{pk}_N, \text{ek}_N))$, or if $\tilde{c}_i \neq c_i$ for some honest $i \in \bar{T}$.

Step 3: The simulator runs the decryption simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ for protocol $\Pi_{\text{DEC}}^{\text{MAL}}$ (interacting with \mathcal{A}^{MAL}). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ queries the ideal decryption functionality with secret key and randomness pairs $\{\tilde{\text{sk}}_i, \tilde{r}_i\}_{i \in T}$, the simulator checks that $\text{Keygen}(1^\kappa; \tilde{r}_i) = (\text{pk}_i, \tilde{\text{sk}}_i, \text{ek}_i)$ for all $i \in T$. If the check fails, it outputs \perp . Otherwise, it decrypts \tilde{c}_i with the secret key $\tilde{\text{sk}}_i$ to obtain the corrupted input \tilde{x}_i (if $\text{Dec}(\tilde{\text{sk}}_i, \tilde{c}_j) = \perp$, it returns \perp):

$$\tilde{x}_i := \text{Dec}(\tilde{\text{sk}}_i, \tilde{c}_j)$$

Finally, it submits inputs $\{\tilde{x}_i\}_{i \in T}$ to the ideal functionality \mathcal{F} , and obtains output $\tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_N)$, where $\tilde{x}_i = x_i$ for honest parties $i \in \bar{T}$. It returns \tilde{y} to the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$.

Output: The simulator receives the output of the corrupted parties from $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$, and returns these as its output.

We prove that $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{MAL}}}(\vec{x}) \stackrel{c}{\approx} \text{REAL}_{\Pi^{\text{MAL}}, \mathcal{A}^{\text{MAL}}}(\vec{x})$ via a hybrid argument.

Hybrid 0: This is a real-world execution of the protocol.

Hybrid 1: We change how Step 3 is performed. Instead of executing the protocol $\Pi_{\text{DEC}}^{\text{MAL}}$ where honest parties use their individual secret keys, we run the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ (interacting with

\mathcal{A}^{MAL}). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ queries the ideal decryption functionality with secret keys and randomness $\{\tilde{\text{sk}}_i, \tilde{r}_i\}_{i \in T}$, we return

$$\tilde{y} = g_{c, \text{pk}_1, \text{ek}_1, \dots, \text{pk}_N, \text{ek}_N} \left((\tilde{\text{sk}}_1, \tilde{r}_1) \dots, (\tilde{\text{sk}}_N, \tilde{r}_N) \right)$$

where $\tilde{\text{sk}}_i = \text{sk}_i$ and $\tilde{r}_i = r_i$ for honest parties $i \in \bar{T}$. We define the output of the corrupted parties to be the output of $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$, and the output of the honest parties to be \tilde{y} .

We claim that Hybrid 0 is computationally indistinguishable from Hybrid 1 by the *security of $\Pi_{\text{DEC}}^{\text{MAL}}$* . Indeed, the security of the decryption protocol $\Pi_{\text{DEC}}^{\text{MAL}}$ guarantees that as long as we correctly emulate the ideal decryption functionality, the joint output of all parties is computationally indistinguishable in a real-world execution of the protocol with adversary \mathcal{A}^{MAL} (Hybrid 0), and in an ideal-world execution of the protocol with adversary $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ (Hybrid 1). We correctly emulate the ideal decryption functionality, by definition.

Hybrid 2: Hybrid 2 is the same as Hybrid 1 except that we use the extractor Ext^Φ to extract a witness $\{(\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}})\}_{i \in [N]}$ from φ_h :

$$\{\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}}\}_{i \in [N]} \leftarrow \text{Ext}^\Phi \left(\{(\text{pk}_i, \text{ek}_i, \text{hk}_i, d_i)\}_{i \in [N]}, \varphi_h \right)$$

We define the output of the protocol to be \perp if for any $i \in [N]$, verification fails for $\tilde{\pi}_i^{\text{ENC}}$ or $d_i \neq H_{\text{hk}_i}(\tilde{c}_i)$. We also output \perp if $c \neq \text{Eval}(C, (\tilde{c}_1, \text{pk}_1, \text{ek}_1), \dots, (\tilde{c}_N, \text{pk}_N, \text{ek}_N))$, where c is the ciphertext returned by \mathcal{A}^{MAL} in Step 2. By the *adaptive extractability* property of Φ , we know that this event happens with negligible probability. Therefore, Hybrid 1 and Hybrid 2 are statistically close.

Note that we require Φ to satisfy *adaptive* extractability because the adversary is free to choose the statement of the proof *after* it sees vrs_h .

Hybrid 3: In Hybrid 3, we additionally let the output of the protocol be \perp if $\tilde{c}_i \neq c_i$ for any honest $i \in \bar{T}$.

We claim that Hybrid 2 and 3 are statistically close by the *collision-resistance* of \mathcal{H} . Indeed, Hybrids 2 and 3 are identical except in the case when all previous checks pass but there exists $j \in \bar{T}$ such that $\tilde{c}_j \neq c_j$. Let ε be the probability, conditioned on all other checks passing, that there exists such a $j \in \bar{T}$. Suppose, for the sake of contradiction, that ε is non-negligible. Then we construct an adversary \mathcal{B} that breaks the collision-resistance of \mathcal{H} . The reduction \mathcal{B} works as follows:

1. The reduction chooses arbitrary inputs $\{x_i\}$.
2. It creates the NIZK CRS honestly: $(\text{crs}^{\text{ENC}}, \cdot) \leftarrow \text{Setup}^{\text{ENC}}(1^\kappa)$, and runs \mathcal{A}^{MAL} on inputs $\{x_i\}_{i \in T}$ and crs^{ENC} as the CRS.
3. For all non-computing parties and honest parties, it samples key tuples $(\text{pk}_i, \text{sk}_i, \text{ek}_i) \leftarrow \text{Keygen}(1^\kappa)$, and encrypts the input correctly: $c_i \leftarrow \text{Enc}(\text{pk}_i, x_i; s_i)$. It creates honest proofs $\pi_i^{\text{ENC}} \leftarrow \text{Prove}^{\text{ENC}}((\text{pk}_i, c_i), (x_i, s_i))$. It also runs the verification setup honestly to generate a verification reference string $(\text{vrs}_i, \cdot) \leftarrow \text{Setup}^\Phi(1^\kappa)$.

4. When it receives a hash key \mathbf{hk} from the collision-resistance challenger, the reduction guesses an honest index $i^* \leftarrow \bar{T}$ uniformly at random and sets $\mathbf{hk}_{i^*} = \mathbf{hk}$. For all other $i \neq i^*$, it samples \mathbf{hk}_i honestly. Finally, for all non-computing and honest parties, it computes the digest $d_i = H_{\mathbf{hk}_i}(c_i)$.
5. It sends $\{\mathbf{pk}_i, \mathbf{ek}_i, c_i, \pi_i^{\text{ENC}}, \mathbf{hk}_i, d_i\}_{i \in \bar{T}}$ to \mathcal{A}^{MAL} .
6. When it receives a ciphertext c and proofs $\varphi_1, \dots, \varphi_N$ from \mathcal{A}^{MAL} , along with the set $\{\mathbf{pk}_i, \mathbf{ek}_i, \mathbf{hk}_i, d_i\}$, it runs the extractor

$$\{\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}}\}_{i \in [N]} \leftarrow \text{Ext}^\Phi \left(\{(\mathbf{pk}_i, \mathbf{ek}_i, \mathbf{hk}_i, d_i)\}_{i \in [N]}, \varphi_h \right)$$

7. Finally, it submits c_{i^*} and \tilde{c}_{i^*} to the collision-resistance challenger as its collision.

If all previous checks pass, then in both hybrids we have that $H(c_j) = H(\tilde{c}_j) = d_j$. Therefore the probability that \mathcal{B} submits a valid collision to the collision challenger is $\varepsilon/|\bar{T}|$. If ε is non-negligible, then \mathcal{B} breaks the collision-resistance property of the hash family \mathcal{H} .

Hybrid 4: In Hybrid 4, we additionally let the output of the protocol be \perp if $\text{Dec}(\tilde{\mathbf{sk}}_i, \tilde{c}_i) = \perp$ for any corrupt $i \in T$, where $\tilde{\mathbf{sk}}_i$ is the secret key output by the decryption protocol simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ and \tilde{c}_i is extracted from the succinct argument φ_h , as in Hybrids 2 and 3.

We claim that Hybrid 3 and Hybrid 4 are statistically close by the *soundness* of the NIZK Π^{ENC} . Indeed, Hybrids 3 and 4 are identical except in the case when all previous checks pass but there exists $j \in T$ such that $\text{Dec}(\tilde{\mathbf{sk}}_j, \tilde{c}_j) = \perp$. By *correctness of decryption*, this happens if and only if $\nexists (\tilde{x}_j, \tilde{s}_j)$ such that $\text{Enc}(\mathbf{pk}_j, \tilde{x}_j; \tilde{s}_j) = \tilde{c}_j$, or in other words, if $(\mathbf{pk}_j, \tilde{c}_j) \notin L^{\text{ENC}}$. Let ε be the probability, conditioned on all other checks passing, that there exists an index $j \in T$ such that $(\mathbf{pk}_j, \tilde{c}_j) \notin L^{\text{ENC}}$. Suppose, for the sake of contradiction, that ε is non-negligible. Then we construct an adversary \mathcal{B} that breaks the soundness of Π^{ENC} . The reduction \mathcal{B} works as follows:

1. The reduction chooses arbitrary inputs $\{x_i\}$.
2. It receives the CRS from the soundness challenger, and runs \mathcal{A}^{MAL} on inputs $\{x_i\}_{i \in T}$ and the CRS.
3. For all non-computing parties and honest parties, it samples key tuples $(\mathbf{pk}_i, \mathbf{sk}_i, \mathbf{ek}_i) \leftarrow \text{Keygen}(1^\kappa)$, and encrypts the input correctly: $c_i \leftarrow \text{Enc}(\mathbf{pk}_i, x_i; s_i)$. It creates honest proofs $\pi_i^{\text{ENC}} \leftarrow \text{Prove}^{\text{ENC}}((\mathbf{pk}_i, c_i), (x_i, s_i))$. It also runs the verification setup honestly to generate a verification reference string $(\mathbf{vrs}_i, \cdot) \leftarrow \text{Setup}^\Phi(1^\kappa)$.
4. It samples \mathbf{hk}_i honestly and computes the digest $d_i = H_{\mathbf{hk}_i}(c_i)$.
5. It sends $\{\mathbf{pk}_i, \mathbf{ek}_i, c_i, \pi_i^{\text{ENC}}, \mathbf{hk}_i, d_i\}_{i \in \bar{T}}$ to \mathcal{A}^{MAL} .
6. When it receives a ciphertext c and proofs $\varphi_1, \dots, \varphi_N$ from \mathcal{A}^{MAL} , along with the set $\{\mathbf{pk}_i, \mathbf{ek}_i, \mathbf{hk}_i, d_i\}$, it runs the extractor

$$\{\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}}\}_{i \in [N]} \leftarrow \text{Ext}^\Phi \left(\{(\mathbf{pk}_i, \mathbf{ek}_i, \mathbf{hk}_i, d_i)\}_{i \in [N]}, \varphi_h \right)$$

7. It runs the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ (interacting with \mathcal{A}^{MAL}). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ queries the ideal decryption functionality with secret key and randomness pairs $\{\widetilde{\text{sk}}_i, \widetilde{r}_i\}_{i \in T}$, it checks that $(\text{pk}_i, \widetilde{\text{sk}}_i, \text{ek}_i) = \text{Keygen}(1^\kappa; \widetilde{r}_i)$ for all $i \in [N]$. If this check fails, it returns \perp . Otherwise, it chooses a corrupt $i^* \leftarrow T$ uniformly at random and submits $\widetilde{\pi}_{i^*}^{\text{ENC}}$ as its proof forgery.

If all previous checks pass, then in both hybrids we have that $\text{Verify}((\text{pk}_i, \widetilde{c}_i), \widetilde{\pi}_i^{\text{ENC}}) = 1$ for all $i \in [N]$ (see Hybrid 2). Therefore, the probability that \mathcal{B} submits a valid forgery to the soundness challenger is $\varepsilon/|T|$. If ε is non-negligible, then \mathcal{B} breaks the soundness property of the NIZK Π^{ENC} .

Hybrid 5: We now change how we compute \widetilde{y} , the value returned to the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ when it queries the decryption ideal functionality. Instead of computing $\widetilde{y} = g_{c, \text{pk}_1, \text{ek}_1, \dots, \text{pk}_N, \text{ek}_N}((\widetilde{\text{sk}}_1, \widetilde{r}_1) \dots, (\widetilde{\text{sk}}_N, \widetilde{r}_N))$, we first check if $(\text{pk}_i, \widetilde{\text{sk}}_i, \text{ek}_i) = \text{Keygen}(1^\kappa; \widetilde{r}_i)$ for all $i \in T$. If this check fails, we return \perp ; otherwise we decrypt each malicious \widetilde{c}_i and evaluate f on the resulting inputs:

$$\widetilde{y} = \begin{cases} f(\widetilde{x}_1, \dots, \widetilde{x}_N) & \text{If } (\text{pk}_i, \widetilde{\text{sk}}_i, \text{ek}_i) = \text{Keygen}(1^\kappa; \widetilde{r}_i) \quad \forall i \in T \\ \perp & \text{Otherwise} \end{cases}$$

$$\text{where } \widetilde{x}_i := \text{Dec}(\widetilde{\text{sk}}_i, \widetilde{c}_i) \text{ for } i \in T \quad \text{and} \quad \widetilde{x}_i = x_i \text{ for } i \in \overline{T}$$

We claim that Hybrid 5 and Hybrid 4 are statistically close. In the case when $(\text{pk}_i, \widetilde{\text{sk}}_i, \text{ek}_i) \neq \text{Keygen}(1^\kappa; \widetilde{r}_i)$ for some $i \in T$, both hybrids output \perp . We focus on the case when this check passes for all parties, so that $\widetilde{\text{sk}}_i$ is guaranteed to be a *valid* secret key for its corresponding public and evaluation keys. In both hybrids, we know that $c = \text{Eval}(C, (\widetilde{c}_1, \text{pk}_1, \text{ek}_1), \dots, (\widetilde{c}_N, \text{pk}_N, \text{ek}_N))$ (see Hybrid 2). By *soundness* of Π^{ENC} , we know that all \widetilde{c}_i 's are fresh encryptions, so by *correctness of multikey evaluation* we know that $\text{Dec}(\widetilde{\text{sk}}_1, \dots, \widetilde{\text{sk}}_N, c) = f(\widetilde{x}_1, \dots, \widetilde{x}_N)$, where we define $\widetilde{\text{sk}}_i = \text{sk}_i$ for all honest $i \in \overline{T}$ and $\widetilde{x}_i := \text{Dec}(\widetilde{\text{sk}}_i, \widetilde{c}_i)$ for all $i \in [N]$. Furthermore, since $\widetilde{c}_i = c_i$ for all honest $i \in \overline{T}$ (see Hybrid 3), we know that $\widetilde{x}_i = x_i$ for all $i \in \overline{T}$ by *correctness of decryption*.

Hybrid 6: In Hybrid 6, we change how we compute the proofs π_i^{ENC} . Instead of computing real proofs, we use the NIZK simulator to create simulated proofs:

$$\{\pi_i^{\text{ENC}} \leftarrow \text{Sim}(\text{tk}^{\text{ENC}}, (\text{pk}_i, c_i))\}_{i \in \overline{T}}$$

We claim that Hybrid 6 is computationally indistinguishable from Hybrid 5 by the *unbounded zero-knowledge property* of the proof system Π^{ENC} . Suppose, for the sake of contradiction, that there exists an algorithm \mathcal{D} that distinguishes between hybrids 5 and 6. We construct an adversary \mathcal{B} that breaks zero-knowledge of Π^{ENC} . The reduction \mathcal{B} works as follows:

1. The reduction chooses arbitrary inputs $\{x_i\}$.
2. It receives the CRS from the zero-knowledge challenger, and runs \mathcal{A}^{MAL} on inputs $\{x_i\}_{i \in T}$ and the CRS.

3. For all non-computing parties and honest parties, it samples key tuples $(\mathbf{pk}_i, \mathbf{sk}_i, \mathbf{ek}_i) \leftarrow \text{Keygen}(1^\kappa)$, and encrypts the input correctly: $c_i \leftarrow \text{Enc}(\mathbf{pk}_i, x_i ; s_i)$. It creates proofs π_i^{ENC} by calling its oracle with statement (\mathbf{pk}_i, c_i) and witness (x_i, s_i) . It also runs the verification setup honestly to generate a verification reference string $(\text{vrs}_i, \cdot) \leftarrow \text{Setup}^\Phi(1^\kappa)$.
4. It samples \mathbf{hk}_i honestly and computes the digest $d_i = H_{\mathbf{hk}_i}(c_i)$.
5. It sends $\{\mathbf{pk}_i, \mathbf{ek}_i, c_i, \pi_i^{\text{ENC}}, \mathbf{hk}_i, d_i\}_{i \in \bar{T}}$ to \mathcal{A}^{MAL} .
6. When it receives a ciphertext c and proofs $\varphi_1, \dots, \varphi_N$ from \mathcal{A}^{MAL} , along with the set $\{\mathbf{pk}_i, \mathbf{ek}_i, \mathbf{hk}_i, d_i\}$, it runs the extractor

$$\{\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}}\}_{i \in [N]} \leftarrow \text{Ext}^\Phi \left(\{(\mathbf{pk}_i, \mathbf{ek}_i, \mathbf{hk}_i, d_i)\}_{i \in [N]}, \varphi_h \right)$$

7. It runs the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ (interacting with \mathcal{A}^{MAL}). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ queries the ideal decryption functionality with secret key and randomness pairs $\{\tilde{\mathbf{sk}}_i, \tilde{r}_i\}_{i \in T}$, it checks that $(\mathbf{pk}_i, \tilde{\mathbf{sk}}_i, \mathbf{ek}_i) \neq \text{Keygen}(1^\kappa ; \tilde{r}_i)$. If this check fails, it returns \perp ; otherwise it returns $\tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_N)$ where $\tilde{x}_i := \text{Dec}(\tilde{\mathbf{sk}}_i, \tilde{c}_i)$ for $i \in T$ and $\tilde{x}_i = x_i$ for $i \in \bar{T}$.
8. At the end of the protocol, it forwards \mathcal{A}^{MAL} 's output to \mathcal{D} as the output of the corrupt parties, and gives \tilde{y} to \mathcal{D} as the output of the honest parties.

When \mathcal{B} 's oracle is the prover oracle $\mathcal{P}(\cdot)$, then \mathcal{B} perfectly emulates Hybrid 5, whereas if the oracle is the simulation oracle $\mathcal{SZM}_{\text{tk}}(\cdot)$, \mathcal{B} perfectly emulates Hybrid 6. Therefore, if \mathcal{D} can distinguish between Hybrids 5 and 6, then \mathcal{B} breaks the zero-knowledge property of Π^{ENC} .

Hybrids 7.k for $k = 1, \dots, N - t$: Let $\bar{T} = \{i_1, \dots, i_{N-t}\}$. In Hybrid 7.k we change c_{i_k} so that instead of encrypting x_{i_k} it now encrypts 0. More formally, in Hybrid 7.k we have:

$$\left\{ c_{i_j} \leftarrow \text{Enc}(\mathbf{pk}_{i_j}, 0) \right\}_{j \leq k} \quad , \quad \left\{ c_{i_j} \leftarrow \text{Enc}(\mathbf{pk}_{i_j}, x_{i_j}) \right\}_{j > k}$$

For ease of notation we let Hybrid 6 be Hybrid 7.0. We claim that the view of \mathcal{A}^{MAL} in Hybrid 7.k is indistinguishable from its view in Hybrid 7.(k-1) by the *semantic security* of \mathcal{E} under public key \mathbf{pk}_{i_k} . Indeed, now that we run the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ in Step 3 instead of the real decryption protocol, the secret key \mathbf{sk}_{i_k} is only used to encrypt c_{i_k} . So suppose, for the sake of contradiction, that there exists an algorithm \mathcal{D} that distinguishes between hybrids 7.k and 7.(k-1). We construct an adversary \mathcal{B} that breaks the semantic security of \mathcal{E} under public key \mathbf{pk}_{i_k} . The reduction \mathcal{B} works as follows:

1. The reduction chooses arbitrary $\{x_i\}$.
2. It creates the NIZK CRS honestly: $(\text{crs}^{\text{ENC}}, \text{tk}^{\text{ENC}}) \leftarrow \text{Setup}^{\text{ENC}}(1^\kappa)$, and runs \mathcal{A}^{MAL} on inputs $\{x_i\}_{i \in T}$ and crs^{ENC} as the CRS.
3. It receives $(\mathbf{pk}, \mathbf{ek})$ from the semantic security challenger and sets $\mathbf{pk}_{i_k} = \mathbf{pk}$ and $\mathbf{ek}_{i_k} = \mathbf{ek}$. Gives $m_0 = 0$ and $m_1 = x_{i_k}$ to the challenger and receives $c = \text{Enc}(\mathbf{pk}, m_b)$. Sets $c_{i_k} = c$. For all $i \in \bar{T}, i \neq i_k$, computes $(\mathbf{pk}_i, \cdot, \mathbf{ek}_i) \leftarrow \text{Keygen}(1^\kappa)$ honestly. For $j < k$, computes $c_{i_j} \leftarrow \text{Enc}(\mathbf{pk}_{i_j}, 0)$ and for $j > k$, computes $c_{i_j} \leftarrow \text{Enc}(\mathbf{pk}_{i_j}, x_{i_j})$.

4. For all non-computing and honest parties, it creates simulated proofs $\pi_i^{\text{ENC}} \leftarrow \text{Sim}(\text{tk}^{\text{ENC}}, (\text{pk}_i, c_i))$ using the trapdoor tk^{ENC} . It also runs the verification setup honestly to generate a verification reference string $(\text{vrs}_i, \cdot) \leftarrow \text{Setup}^\Phi(1^\kappa)$.
5. It samples hk_i honestly and computes the digest $d_i = H_{\text{hk}_i}(c_i)$.
6. It sends $\{\text{pk}_i, \text{ek}_i, c_i, \pi_i^{\text{ENC}}, \text{hk}_i, d_i\}_{i \in \overline{T}}$ to \mathcal{A}^{MAL} .
7. When it receives a ciphertext c and proofs $\varphi_1, \dots, \varphi_N$ from \mathcal{A}^{MAL} , along with the set $\{\text{pk}_i, \text{ek}_i, \text{hk}_i, d_i\}$, it runs the extractor

$$\{\tilde{c}_i, \tilde{\pi}_i^{\text{ENC}}\}_{i \in [N]} \leftarrow \text{Ext}^\Phi \left(\{(\text{pk}_i, \text{ek}_i, \text{hk}_i, d_i)\}_{i \in [N]}, \varphi_h \right)$$

8. It runs the simulator $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ (interacting with \mathcal{A}^{MAL}). When $\mathcal{S}_{\Pi_{\text{DEC}}}^{\text{MAL}}$ queries the ideal decryption functionality with secret key and randomness pairs $\{\tilde{\text{sk}}_i, \tilde{r}_i\}_{i \in T}$, it checks that $(\text{pk}_i, \tilde{\text{sk}}_i, \text{ek}_i) \neq \text{Keygen}(1^\kappa; \tilde{r}_i)$. If this check fails, it returns \perp ; otherwise it returns $\tilde{y} = f(\tilde{x}_1, \dots, \tilde{x}_N)$ where $\tilde{x}_i := \text{Dec}(\tilde{\text{sk}}_i, \tilde{c}_i)$ for $i \in T$ and $\tilde{x}_i = x_i$ for $i \in \overline{T}$.
9. At the end of the protocol, it forwards \mathcal{A}^{MAL} 's output to \mathcal{D} as the output of the corrupt parties, and gives \tilde{y} to \mathcal{D} as the output of the honest parties.

When $b = 0$, \mathcal{B} perfectly emulates Hybrid 7. k , whereas if $b = 1$, \mathcal{B} perfectly emulates Hybrid 7. $(k - 1)$. Therefore, if \mathcal{D} can distinguish between Hybrids 7. k and 7. $(k - 1)$, then \mathcal{B} can distinguish between an encryption of m_0 and an encryption of m_1 , contradicting the semantic security of \mathcal{E} .

We have proved that the joint output in Hybrid 0 is computationally indistinguishable from the joint output in Hybrid 7. $(N - t)$. Notice that the joint output in Hybrid 7. $(N - t)$ is precisely $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{MAL}}}(\vec{x})$, and the joint output in Hybrid 0 is defined to be $\text{REAL}_{\Pi^{\text{SM}}, \mathcal{A}^{\text{MAL}}}(\vec{x})$. We conclude that $\text{IDEAL}_{\mathcal{F}, \mathcal{S}^{\text{MAL}}}(\vec{x}) \approx \text{REAL}_{\Pi^{\text{SM}}, \mathcal{A}^{\text{MAL}}}(\vec{x})$, as desired. \square

4.2.3 Efficient NIZKs to Prove Plaintext Knowledge

The protocol described in Section 4.2.1 requires a NIZK argument system for the relation NP relation $R^{\text{ENC}} = \{ ((\text{pk}, c), (x, s)) \mid c = \text{Enc}(\text{pk}, x; s) \}$. While it is known how to construct NIZK argument systems for all of NP [GOS06, GOS12], using this construction requires expensive NP reductions. In this section, we show how to construct an efficient gap Σ -protocol for R^{ENC} when the encryption scheme is the NTRU-based multikey FHE scheme from Section 3.4. By Theorem 2.2 this suffices to construct an efficient NIZK argument system for R^{ENC} in the random oracle model. Our construction follows the ideas of Asharov et al. [AJW11, AJL⁺12].

Recall that in the aforementioned FHE scheme, a ciphertext has the form $c = [hs + 2e + m]_q$ for public key h , message $m \in \{0, 1\}$, and ring elements s, e , sampled from B -bounded distribution χ . We construct a gap Σ -protocol for proving that “ c encrypts 0 under h ”. That is, we show a protocol for relation

$$R_0^{\text{ENC}} = \left\{ ((h, c), (s, e)) \mid c = [hs + 2e]_q \wedge \|s\|_\infty, \|e\|_\infty \leq B \right\}$$

with corresponding language L_0^{ENC} . By Theorem 2.1, we can then construct a gap Σ -protocol for R^{ENC} using an OR protocol to prove that “ $c \in L_0^{\text{ENC}}$ or $c - 1 \in L_0^{\text{ENC}}$ ”.

Gap Σ -protocol for Encryptions of 0. Our construction of a gap Σ -protocol for R_0^{ENC} uses the same parameters as the encryption scheme: degree n , polynomial $\phi(x) = x^n + 1$, modulus q , and distribution $\chi = \overline{D}_{\mathbb{Z}^n, r}$ over the ring $R = \mathbb{Z}[x]/\langle\phi(x)\rangle$. It is additionally parametrized by a distribution $\tilde{\chi} = \overline{D}_{\mathbb{Z}^n, \tilde{r}}$ over R , such that $2^{\omega(\log \kappa)}r \leq \tilde{r} \leq q/4\sqrt{n} - r$. To simplify notation, we recall from Lemma 2.7 that χ is B -bounded and $\tilde{\chi}$ is \tilde{B} -bounded for $B = r\sqrt{n}$ and $\tilde{B} = \tilde{r}\sqrt{n}$. By our choice of \tilde{r} , this means that $\tilde{B} + B \leq q/4$.

To formally describe our protocol, we must first define relations R_{zk} and R_{sound} . We set $B_{\text{zk}} = R_0^{\text{ENC}}$ and set B_{sound} to be essentially the same as R_0^{ENC} , differing only in the requirement set for $\|s\|_\infty$ and $\|e\|_\infty$:

$$R_{\text{sound}} = \left\{ ((h, c), (s, e)) \mid c = [hs + 2e]_q \wedge \|s\|_\infty, \|e\|_\infty \leq 4(\tilde{B} + B) \right\}$$

Note that since $\tilde{B} \geq B$, we have $R_{\text{zk}} \subseteq R_{\text{sound}}$. We can now describe our construction:

- $P_1((h, c), (s, e))$: Samples $\tilde{s}, \tilde{e} \leftarrow \tilde{\chi}$ and outputs $a = [h\tilde{s} + 2\tilde{e}]_q$ and $st = (\tilde{s}, s)$.
- $V_1((h, c))$: Outputs a random bit $b \leftarrow \{0, 1\}$.
- $P_2(st, b)$: Parses $st = (\tilde{s}, s)$ and outputs $z = [\tilde{s} + bs]_q$.
- $V_2((h, c), a, b, z)$: Computes $\varepsilon = [(a + bc) - hz]_q$ and outputs 1 if and only if $\|z\|_\infty \leq \tilde{B} + B$, $\|\varepsilon\|_\infty \leq 2(\tilde{B} + B)$, and ε is even.

Theorem 4.3. *Let $R_{\text{zk}}, R_{\text{sound}}$ be the NP relations described above. The construction $\langle P, V \rangle$ with $P = (P_1, P_2)$ and $V = (V_1, V_2)$ is a gap Σ -protocol for $(R_{\text{zk}}, R_{\text{sound}})$.*

Proof. We show that the above construction satisfies the completeness, special soundness, and HVZK properties.

Completeness: Let $((h, c), (s, e)) \in L_{\text{zk}}$, and let (a, b, z) be a transcript for protocol $\langle P, V \rangle$. Then

$$\varepsilon = [(a + bc) - hz]_q = [h\tilde{s} + 2\tilde{e} + bhs + 2be - h\tilde{s} - hbs]_q = [2(\tilde{e} + be)]_q = 2(\tilde{e} + be)$$

where the last inequality holds by the fact that $\tilde{B} + B \leq q/4$. It is clear that ε is even, and its coefficients are bounded by $2(\tilde{B} + B)$. Furthermore, $z = \tilde{s} + bs$, so $\|z\|_\infty \leq \tilde{B} + B$, as required.

Special Soundness: Let (h, c) be a public key and ciphertext pair, and let $(a, 0, z_0)$ and $(a, 1, z_1)$ be two accepting transcripts. The extractor Ext outputs (s^*, e^*) , where $s^* = z_1 - z_0$ and $e^* = [c - hs^*]_q$.

We now argue that $((h, c), (s^*, e^*)) \in R_{\text{sound}}$. By construction, we have that $c = [hs^* + 2e^*]_q$. It remains to show the bound on the size of the coefficients of s^* and e^* . Since $(a, 0, z_0)$ and $(a, 1, z_1)$ are accepting transcripts, we know that $\|z_0\|_\infty, \|z_1\|_\infty \leq \tilde{B} + B$, so that $\|s^*\|_\infty \leq 2(\tilde{B} + B)$.

We now bound e^* . Let $\varepsilon_0 = [a - hz_0]_q$ and $\varepsilon_1 = [(a + c) - hz_1]_q$. Since $(a, 0, z_0)$ and $(a, 1, z_1)$ are accepting transcripts, we know that $\|\varepsilon_0\|_\infty, \|\varepsilon_1\|_\infty \leq 2(\tilde{B} + B)$ and both ε_0 and ε_1 are

even. Furthermore, $\varepsilon_1 - \varepsilon_0 = [(a + c) - hz_1 - (a - hz_0)]_q = [c - h(z_1 - z_0)]_q = e^*$. This means that e^* is even since both ε_0 and ε_1 are even, and we also have that $\|e^*\|_\infty \leq \|\varepsilon_0\|_\infty + \|\varepsilon_1\|_\infty \leq 4(\tilde{B} + B)$, as desired.

Honest-Verifier Zero-Knowledge: Let $((h, c), (s, e)) \in L_{\text{zk}}$ and let $b \in \{0, 1\}$. The simulator Sim chooses $z', e' \leftarrow \tilde{\chi}$, sets $a' = hz' + 2e' + bc$, and outputs (a', b, z') . We argue that the output of Sim is statistically close to the transcript (a, b, z) of an execution of the protocol $\langle P, V \rangle$. In a real transcript, we have $a = h\tilde{s} + 2\tilde{e}$ and $z = \tilde{s} + \sigma s$. In the simulated transcript, we have $a' = h(z' + bs) + 2(e' + be)$. If $b = 0$, then the distributions are *identical* because $\tilde{s}, \tilde{e}, z', e'$ are all sampled from the same distribution $\tilde{\chi}$. On the other hand, if $b = 1$, then the distributions are *statistically close* by Corollary 2.9.

□

Consequences of Having a Gap. We have shown how to construct efficient NIZK arguments for the relation R^{ENC} for the NTRU-based multikey FHE scheme from Section 3.4. However, there is a *gap* in the relations for which soundness and zero-knowledge hold: zero-knowledge holds for an honest prover with a statement in R_{zk} , but an honest verifier is only convinced that the statement is in $R_{\text{sound}} \supseteq R_{\text{zk}}$. We must show that this gap does not affect the correctness of our protocol. It suffices to prove that the scheme is fully homomorphic when the error in fresh ciphertexts is bounded by $B^* \stackrel{\text{def}}{=} 4(\tilde{B} + B)$.

Our analysis in Section 3.4 does not immediately guarantee this, as it sets $B = \text{poly}(n)$. Since we must have $n = \text{poly}(\kappa)$ for efficiency of the scheme, this means $B = \text{poly}(\kappa)$. However B^* is super-polynomial in κ . Nevertheless, we can easily modify our parameters and analysis to guarantee that the scheme remains fully homomorphic with ciphertext noise that is super-polynomial in κ .

The proof of Lemma 3.6 shows that the leveled homomorphic scheme \mathcal{E}_{LH} described in Section 3.4.2 is multikey homomorphic for N keys and circuits of depth D as long as

$$(nB^*)^{2N+2} < \frac{2^{n^\varepsilon}}{2(8n(nB^*)^{2N+2})^D}$$

which yields the requirement $ND = O(n^\varepsilon/(\log n + \log B^*))$. We can then follow the proof of Theorem 3.9 and show that there exists a multikey fully homomorphic encryption scheme for $N = O\left(\sqrt{(n^\varepsilon/\log n(\log n + \log B^*))}\right)$. If we set $\tilde{B} = 2^{\log^2 \kappa} \cdot B$ for $B = \text{poly}(n)$ and $n \geq \kappa$, this is guaranteed if $N = O\left(\sqrt{(n^\varepsilon/\log^3 n)}\right)$ since

$$n^\varepsilon/(\log^3 n) = O(n^\varepsilon/(\log n \cdot (\log n + \log^2 \kappa))) = O(n^\varepsilon/(\log n \cdot (\log n + \log B^*)))$$

(In)Security in the Standard Model. We have shown a NIZK argument for relation R^{ENC} . Though secure in the random oracle model, we remark that care must be taken if we want to hope for security in the standard model. More specifically, since our gap Σ -protocol has only constant soundness, we need to use parallel repetition for soundness amplification. For efficiency, we would like to repeat the protocol only $\text{polylog}(\kappa)$ many times as this already achieves negligible soundness. However, Dachman-Soled et al. [DJKL12, BDG⁺13] have shown that if we use such a small number of repetitions, the resulting NIZK cannot be proven sound (in the standard model)

via a black-box reduction to a (super-polynomially hard) falsifiable assumption. Also see remarks after Theorem 2.2.

4.3 Impossibility of a 2-Round Protocol

We have shown that there exists an on-the-fly MPC protocol with a 5-round online phase. We now ask whether we can achieve the optimal solution of having a completely non-interactive online phase. In this section we answer this question negatively: we show that the existence of such a protocol (secure against semi-honest adversaries)¹⁴ implies general circuit obfuscation as a virtual black-box with single-bit output, which we know to be impossible [BGI⁺01]. Our techniques are inspired by those of van Dijk and Jules [vDJ10].

We begin by reviewing the definition of general circuit obfuscation [BGI⁺01].

Definition 4.1 (Circuit Obfuscation [BGI⁺01]). *A probabilistic algorithm \mathcal{O} is a circuit obfuscator if the following three conditions hold:*

Functionality: *For every circuit C , the string $\mathcal{O}(C)$ describes a circuit that computes the same function as C .*

Polynomial Slowdown: *There is a polynomial p such that for every circuit C , $|\mathcal{O}(C)| \leq p(|C|)$.*

“Virtual Black-Box” Property: *For any PPT adversary \mathcal{A} , there is a PPT simulator \mathcal{S} such that for all circuits C*

$$\left| \Pr[\mathcal{A}(\mathcal{O}(C)) = 1] - \Pr[\mathcal{S}^C(1^{|C|}) = 1] \right| \leq \text{negl}(|C|)$$

Barak et al. [BGI⁺01] show that assuming one-way functions exist, there does not exist any algorithm \mathcal{O} satisfying Definition 4.1, even if we do not require that \mathcal{O} run in polynomial time. Thus, our results imply that assuming one-way functions exist, there does not exist any on-the-fly MPC protocol with a non-interactive online phase.

We now show the connection between on-the-fly MPC and obfuscation. We consider an on-the-fly MPC protocol with a non-interactive online phase, and assume that only one function is evaluated and the function is chosen a-priori, before the start of the protocol (i.e. it does not depend on the offline stage messages). Let N be the number of inputs of the circuit; without loss of generality, we assume that the computing parties are P_1, \dots, P_N . Note that considering such a restricted protocol only makes our impossibility result stronger. A protocol like this can be modeled by efficient and possibly randomized algorithms: $\text{In}_1, \dots, \text{In}_U, \text{Compute}, \text{Out}_1, \dots, \text{Out}_N$, where:

- $(d_i, c_i) \leftarrow \text{In}_i(x_i)$: On input x_i , the algorithm In_i outputs two elements, c_i to be sent to the server S and d_i to be kept by party P_i .
- $(z_1, \dots, z_N) \leftarrow \text{Compute}(C, c_1, \dots, c_N)$: On input a circuit C and c_1, \dots, c_N , which are the messages the server received from parties P_1, \dots, P_N , Compute outputs N elements z_1, \dots, z_N . The server sends back z_i to party P_i .

¹⁴Considering semi-honest adversaries instead of semi-malicious or malicious adversaries only makes our result stronger.

- $y \leftarrow \text{Out}_i(z_i, d_i)$: On input z_i which was received from the server, and the auxiliary information d_i output by In_i , Out_i computes the output y .

We know from the work of Halevi, Lindell, and Pinkas [HLP11] that in the non-interactive setting, the server can always evaluate the circuit multiple times, keeping some parties inputs but plugging in fake inputs of its choosing for the other parties. Thus we must relax the definition of security so that when the server is corrupted, the simulator is allowed to submit queries of the form (S, \vec{x}) , where S is a non-empty subset of the honest parties and \vec{x} is any input vector of size $n - |S|$. The trusted functionality evaluates the function on \vec{x} and the honest inputs in S . Furthermore, our result holds even when the real-world adversary is only allowed to output 1 bit.¹⁵

Theorem 4.4. *If there exists an on-the-fly MPC protocol with a non-interactive online phase that computes all efficiently computable functions with 2 inputs, and is secure against semi-honest adversaries (with the relaxed definition of security), then there exists a circuit obfuscator \mathcal{O} satisfying Definition 4.1.*

Proof. We start by defining a family of “meta-circuits” $\{F^{(m)}\}_{m \in \mathbb{N}}$. For a fixed $m \in \mathbb{N}$, $F^{(m)}$ is such that given a circuit C of size m and bit-string x , it evaluates C on x and outputs $C(x)$, i.e. $F^{(m)}(C, x) = C(x)$. van Dijk and Juels [vDJ10] show to construct a family of meta-circuits such that for all $m \in \mathbb{N}$, $|F^{(m)}| = O(m^2)$.

We now show how to construct a circuit obfuscator \mathcal{O} using an on-the-fly MPC protocol $\Pi = (\text{In}_1, \dots, \text{In}_U, \text{Compute}, \text{Out}_1, \text{Out}_2)$ with the properties described in the theorem statement. Given a circuit C of size m , \mathcal{O} computes $(\cdot, c_1) \leftarrow \text{In}_1(C)$, samples random coins ρ, σ, τ , and outputs a circuit G that on input x :

- Computes $(c_2, d_2) := \text{In}_2(x ; \rho)$.
- Computes $(\cdot, z_2) := \text{Compute}(F^{(m)}, c_1, c_2 ; \sigma)$
- Computes and outputs $y := \text{Out}_2(z_2, d_2 ; \tau)$.

We now show that this obfuscator satisfies the functionality, polynomial slowdown, and virtual black-box properties from Definition 4.1.

Functionality: The correctness property of the on-the-fly MPC protocol guarantees that $G(x) = F^{(m)}(C, x) = C(x)$ for all x .

Polynomial Slowdown: Using van Dijk and Juel’s construction [vDJ10], we have that $|F^{(m)}| = O(m^2)$. Since all algorithms of the on-the-fly MPC protocol run in polynomial time, we have that there exists a polynomial p such that $|G| = p(|C|)$.

Virtual Black-Box: To prove the virtual black-box property, we observe that given an attacker \mathcal{A} trying to break the obfuscation, we can construct a real-world semi-honest adversary \mathcal{B} attacking the on-the-fly MPC protocol, corrupting the server and party P_2 . The honest party receives input C and \mathcal{B} receives a dummy value \tilde{x} for P_2 , which it ignores. Instead it receives c_1 from the honest party, builds G as specified and runs \mathcal{A} on G . When \mathcal{A} outputs a bit b , \mathcal{B}

¹⁵Considering a restricted class of adversaries for the on-the-fly MPC protocol only makes our impossibility result stronger.

completes Steps 2 and 3 in the protocol as specified, and outputs b . We emphasize that any action taken by \mathcal{A} is valid for a semi-honest adversary, so \mathcal{B} is semi-honest.

Security of Π says that there exists simulator \mathcal{S} such that for all inputs C, \tilde{x} , we have $\text{IDEAL}_{\mathcal{F}, \mathcal{S}}(C, \tilde{x}) \stackrel{c}{\approx} \text{REAL}_{\Pi, \mathcal{B}}(C, \tilde{x})$, where in the ideal world, \mathcal{S} is given access to an oracle as described above. In the setting we are considering, the only valid subset that \mathcal{S} can provide in a query to this oracle is $\{1\}$. Thus, \mathcal{S} has oracle access to $F^{(m)}(C, \cdot) = C(\cdot)$. We can build a simulator \mathcal{S}' with oracle access to $C(\cdot)$ that on input $|C|$ ¹⁶, chooses an arbitrary \tilde{x} and runs $\mathcal{S}(\tilde{x})$ (which runs \mathcal{B} , which runs \mathcal{A}), answers \mathcal{S} 's queries with its own oracle, and outputs \mathcal{S} 's output.

Since \mathcal{B} outputs whatever \mathcal{A} outputs and \mathcal{S}' outputs whatever \mathcal{S} outputs, the fact that $\text{IDEAL}_{\mathcal{F}, \mathcal{S}}(C, \tilde{x}) \stackrel{c}{\approx} \text{REAL}_{\Pi, \mathcal{B}}(C, \tilde{x})$ implies that $\mathcal{S}'(|C|) \stackrel{c}{\approx} \mathcal{A}(G)$. The theorem statement follows. □

References

- [ACPS09] Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 595–618. Springer, 2009.
- [AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In *STOC*, pages 284–293, 1997.
- [AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *ICALP (1)*, volume 6198 of *Lecture Notes in Computer Science*, pages 152–163. Springer, 2010.
- [AJL⁺12] Gilad Asharov, Abhishek Jain, Adriana López-Alt, Eran Tromer, Vinod Vaikuntanathan, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In Pointcheval and Johansson [PJ12], pages 483–501.
- [AJW11] Gilad Asharov, Abhishek Jain, and Daniel Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. *IACR Cryptology ePrint Archive*, 2011:613, 2011.
- [Bar01] Boaz Barak. How to go beyond the black-box simulation barrier. In *FOCS*, pages 106–115. IEEE Computer Society, 2001.
- [BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable

¹⁶In most applications it is ok to leak the size of the honest input. Indeed this is implied in most constructions, including our construction from Section 4.1.

collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, 2012.

- [BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *STOC*, pages 111–120. ACM, 2013.
- [BD10] Rikke Bendlin and Ivan Damgård. Threshold decryption and zero-knowledge proofs for lattice-based cryptosystems. In Daniele Micciancio, editor, *TCC*, volume 5978 of *Lecture Notes in Computer Science*, pages 201–218. Springer, 2010.
- [BDG⁺13] Nir Bitansky, Dana Dachman-Soled, Sanjam Garg, Abhishek Jain, Yael Tauman Kalai, Adriana López-Alt, and Daniel Wichs. Why “fiat-shamir for proofs” lacks a proof. In *TCC*, pages 182–201, 2013.
- [BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Paterson [Pat11], pages 169–188.
- [BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In Douglas R. Stinson, editor, *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 278–291. Springer, 1993.
- [BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In Janos Simon, editor, *STOC*, pages 103–112. ACM, 1988.
- [BGI⁺01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Kilian [Kil01], pages 1–18.
- [BGI⁺12] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. *J. ACM*, 59(2):6, 2012.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.
- [BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.
- [BLLN13] Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved security for a ring-based fully homomorphic encryption scheme. In Martijn Stam, editor, *IMA Int. Conf.*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2013.
- [BLV06] Boaz Barak, Yehuda Lindell, and Salil P. Vadhan. Lower bounds for non-black-box

zero knowledge. *J. Comput. Syst. Sci.*, 72(2):321–391, 2006.

- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.
- [Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In Safavi-Naini and Canetti [SNC12], pages 868–886.
- [BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In Ostrovsky [Ost11], pages 97–106.
- [BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-lwe and security for key dependent messages. In Rogaway [Rog11], pages 505–524.
- [BV14] Zvika Brakerski and Vinod Vaikuntanathan. Lattice-based fhe as secure as pke. In Moni Naor, editor, *ITCS*, pages 1–12. ACM, 2014.
- [CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.
- [CCK⁺13] Jung Hee Cheon, Jean-Sébastien Coron, Jinsu Kim, Moon Sung Lee, Tancrede Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch fully homomorphic encryption over the integers. In Johansson and Nguyen [JN13], pages 315–335.
- [CDN01] Ronald Cramer, Ivan Damgård, and Jesper Buus Nielsen. Multiparty computation from threshold homomorphic encryption. In *EUROCRYPT*, pages 280–299, 2001.
- [CDS94] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *CRYPTO*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.
- [CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.
- [CKV10] Kai-Min Chung, Yael Tauman Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In Rabin [Rab10], pages 483–501.
- [CL08] Giovanni Di Crescenzo and Helger Lipmaa. Succinct np proofs from an extractability assumption. In Arnold Beckmann, Costas Dimitracopoulos, and Benedikt Löwe, editors, *CiE*, volume 5028 of *Lecture Notes in Computer Science*, pages 175–185. Springer, 2008.
- [CLO⁺13] Ashish Choudhury, Jake Loftus, Emmanuela Orsini, Arpita Patra, and Nigel P. Smart. Between a rock and a hard place: Interpolating between mpc and fhe. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (2)*, volume 8270 of *Lecture Notes in Computer Science*, pages 221–240. Springer, 2013.
- [CLT14] Jean-Sébastien Coron, Tancrede Lepoint, and Mehdi Tibouchi. Scale-invariant fully

- homomorphic encryption over the integers. In Hugo Krawczyk, editor, *Public Key Cryptography*, volume 8383 of *Lecture Notes in Computer Science*, pages 311–328. Springer, 2014.
- [CMNT11] Jean-Sébastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully homomorphic encryption over the integers with shorter public keys. In Rogaway [Rog11], pages 487–504.
- [CNT12] Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public key compression and modulus switching for fully homomorphic encryption over the integers. In Pointcheval and Johansson [PJ12], pages 446–464.
- [Cra12] Ronald Cramer, editor. *Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings*, volume 7194 of *Lecture Notes in Computer Science*. Springer, 2012.
- [DIK⁺08] Ivan Damgård, Yuval Ishai, Mikkel Krøigaard, Jesper Buus Nielsen, and Adam Smith. Scalable multiparty computation with nearly optimal work and resilience. In David Wagner, editor, *CRYPTO*, volume 5157 of *Lecture Notes in Computer Science*, pages 241–261. Springer, 2008.
- [DIK10] Ivan Damgård, Yuval Ishai, and Mikkel Krøigaard. Perfectly secure multiparty computation and the computational overhead of cryptography. In Gilbert [Gil10], pages 445–465.
- [DJKL12] Dana Dachman-Soled, Abhishek Jain, Yael Tauman Kalai, and Adriana López-Alt. On the (in)security of the fiat-shamir paradigm, revisited. *IACR Cryptology ePrint Archive*, 2012:706, 2012.
- [DKL⁺13] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure mpc for dishonest majority - or: Breaking the spdz limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2013.
- [DNRS03] Cynthia Dwork, Moni Naor, Omer Reingold, and Larry J. Stockmeyer. Magic functions. *J. ACM*, 50(6):852–921, 2003.
- [DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Safavi-Naini and Canetti [SNC12], pages 643–662.
- [DRV12] Yevgeniy Dodis, Thomas Ristenpart, and Salil P. Vadhan. Randomness condensers for efficiently samplable, seed-dependent sources. In Cramer [Cra12], pages 618–635.
- [FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.

- [FV12] Junfeng Fan and Frederik Vercauteren. Somewhat practical fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2012:144, 2012.
- [Gam84] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *CRYPTO*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18. Springer, 1984.
- [Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.
- [Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *STOC*, pages 169–178. ACM, 2009.
- [GGH97] Oded Goldreich, Shafi Goldwasser, and Shai Halevi. Public-key cryptosystems from lattice reduction problems. In Burton S. Kaliski Jr., editor, *CRYPTO*, volume 1294 of *Lecture Notes in Computer Science*, pages 112–131. Springer, 1997.
- [GGH13a] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Johansson and Nguyen [JN13], pages 1–17.
- [GGH⁺13b] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, pages 40–49. IEEE Computer Society, 2013.
- [GGHR14] Sanjam Garg, Craig Gentry, Shai Halevi, and Mariana Raykova. Two-round secure mpc from indistinguishability obfuscation. In Yehuda Lindell, editor, *TCC*, volume 8349 of *Lecture Notes in Computer Science*, pages 74–94. Springer, 2014.
- [GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Rabin [Rab10], pages 465–482.
- [GGPR13] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In Johansson and Nguyen [JN13], pages 626–645.
- [GH11a] Craig Gentry and Shai Halevi. Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Ostrovsky [Ost11], pages 107–109.
- [GH11b] Craig Gentry and Shai Halevi. Implementing gentry’s fully-homomorphic encryption scheme. In Paterson [Pat11], pages 129–148.
- [GHL⁺11] Craig Gentry, Shai Halevi, Vadim Lyubashevsky, Christopher Peikert, Joseph Silverman, and Nigel Smart. Personal communication, 2011.
- [GHPS12] Craig Gentry, Shai Halevi, Chris Peikert, and Nigel P. Smart. Ring switching in bgv-style homomorphic encryption. In Ivan Visconti and Roberto De Prisco, editors, *SCN*, volume 7485 of *Lecture Notes in Computer Science*, pages 19–37. Springer, 2012.
- [GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Better bootstrapping in fully homomorphic encryption. In Marc Fischlin, Johannes Buchmann, and Mark Manulis,

- editors, *Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2012.
- [GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In Pointcheval and Johansson [PJ12], pages 465–482.
- [GHS12c] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the aes circuit. In Safavi-Naini and Canetti [SNC12], pages 850–867.
- [Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.
- [GK03] Shafi Goldwasser and Yael Tauman Kalai. On the (in)security of the fiat-shamir paradigm. In *FOCS*, pages 102–113. IEEE Computer Society, 2003.
- [GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In Cynthia Dwork, editor, *STOC*, pages 113–122. ACM, 2008.
- [GLR11] Shafi Goldwasser, Huijia Lin, and Aviadi Rubinfeld. Delegation of computation without rejection problem from designated verifier cs-proofs. *Cryptology ePrint Archive: Report 2011/456*, 2011.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.
- [GO94] Oded Goldreich and Yair Oren. Definitions and properties of zero-knowledge proof systems. *J. Cryptology*, 7(1):1–32, 1994.
- [Gol04] Oded Goldreich. *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge University Press, New York, NY, USA, 2004.
- [GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for np. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 339–358. Springer, 2006.
- [GOS12] Jens Groth, Rafail Ostrovsky, and Amit Sahai. New techniques for noninteractive zero-knowledge. *J. ACM*, 59(3):11, 2012.
- [Gro10] Jens Groth. Short pairing-based non-interactive zero-knowledge arguments. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 321–340. Springer, 2010.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *CRYPTO (1)*, volume 8042 of *Lecture Notes in*

Computer Science, pages 75–92. Springer, 2013.

- [GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 99–108. ACM, 2011.
- [HLP11] Shai Halevi, Yehuda Lindell, and Benny Pinkas. Secure computation on the web: Computing without simultaneous interaction. In Rogaway [Rog11], pages 132–150.
- [HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. Ntru: A ring-based public key cryptosystem. In Joe Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 267–288. Springer, 1998.
- [HT98] Satoshi Hada and Toshiaki Tanaka. On the existence of 3-round zero-knowledge protocols. In Hugo Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 408–423. Springer, 1998.
- [JN13] Thomas Johansson and Phong Q. Nguyen, editors. *Advances in Cryptology - EURO-CRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*. Springer, 2013.
- [Kil92] Joe Kilian. A note on efficient zero-knowledge proofs and arguments (extended abstract). In *STOC*, pages 723–732. ACM, 1992.
- [Kil95] Joe Kilian. Improved efficient arguments (preliminary version). In Don Coppersmith, editor, *CRYPTO*, volume 963 of *Lecture Notes in Computer Science*, pages 311–324. Springer, 1995.
- [Kil01] Joe Kilian, editor. *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*. Springer, 2001.
- [KMR11] Seny Kamara, Payman Mohassel, and Mariana Raykova. Outsourcing multi-party computation. Cryptology ePrint Archive, Report 2011/272, 2011. <http://eprint.iacr.org/>.
- [Lip12] Helger Lipmaa. Progression-free sets and sublinear pairing-based non-interactive zero-knowledge arguments. In Cramer [Cra12], pages 169–189.
- [Lip13] Helger Lipmaa. Succinct non-interactive zero knowledge arguments from span programs and linear error-correcting codes. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT (1)*, volume 8269 of *Lecture Notes in Computer Science*, pages 41–60. Springer, 2013.
- [LPR10] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. On ideal lattices and learning with errors over rings. In Gilbert [Gil10], pages 1–23.
- [LTV11] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. Cloud-assisted mul-

- tiparty computation from fully homomorphic encryption. *IACR Cryptology ePrint Archive*, 2011:663, 2011.
- [LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *STOC*, pages 1219–1234. ACM, 2012.
 - [Mic94] Silvio Micali. Cs proofs (extended abstracts). In *FOCS*, pages 436–453. IEEE, 1994.
 - [MSS13] Steven Myers, Mona Sergi, and Abhi Shelat. Black-box proof of knowledge of plaintext and multiparty computation with low communication overhead. In *TCC*, pages 397–417, 2013.
 - [Nao03] Moni Naor. On cryptographic assumptions and challenges. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 96–109. Springer, 2003.
 - [Ost11] Rafail Ostrovsky, editor. *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*. IEEE, 2011.
 - [Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 1999.
 - [Pat11] Kenneth G. Paterson, editor. *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*. Springer, 2011.
 - [PHGR13] Bryan Parno, Jon Howell, Craig Gentry, and Mariana Raykova. Pinocchio: Nearly practical verifiable computation. In *IEEE Symposium on Security and Privacy*, pages 238–252. IEEE Computer Society, 2013.
 - [PJ12] David Pointcheval and Thomas Johansson, editors. *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*. Springer, 2012.
 - [Rab10] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.
 - [RAD78] R L Rivest, L Adleman, and M L Dertouzos. On data banks and privacy homomorphisms. *Foundations of Secure Computation, Academia Press*, pages 169–179, 1978.
 - [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005.
 - [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography.

J. ACM, 56(6), 2009.

- [Rog11] Phillip Rogaway, editor. *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*. Springer, 2011.
- [RTSS09] Thomas Ristenpart, Eran Tromer, Hovav Shacham, and Stefan Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *ACM Conference on Computer and Communications Security*, pages 199–212, 2009.
- [SCO⁺01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In Kilian [Kil01], pages 566–598.
- [SNC12] Reihaneh Safavi-Naini and Ran Canetti, editors. *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*. Springer, 2012.
- [SS11a] Peter Scholl and Nigel P. Smart. Improved key generation for gentry’s fully homomorphic encryption scheme. In Liqun Chen, editor, *IMA Int. Conf.*, volume 7089 of *Lecture Notes in Computer Science*, pages 10–22. Springer, 2011.
- [SS11b] Damien Stehlé and Ron Steinfeld. Making ntru as secure as worst-case problems over ideal lattices. In Paterson [Pat11], pages 27–47.
- [SV10] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Phong Q. Nguyen and David Pointcheval, editors, *Public Key Cryptography*, volume 6056 of *Lecture Notes in Computer Science*, pages 420–443. Springer, 2010.
- [SV14] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic simd operations. *Des. Codes Cryptography*, 71(1):57–81, 2014.
- [Val08] Paul Valiant. Incrementally verifiable computation or proofs of knowledge imply time/space efficiency. In Ran Canetti, editor, *TCC*, volume 4948 of *Lecture Notes in Computer Science*, pages 1–18. Springer, 2008.
- [vDGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In Gilbert [Gil10], pages 24–43.
- [vDJ10] Marten van Dijk and Ari Juels. On the impossibility of cryptography alone for privacy-preserving cloud computing. In *Proceedings of the 5th USENIX conference on Hot topics in security*, HotSec’10, pages 1–8, Berkeley, CA, USA, 2010. USENIX Association.
- [Yao82] Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *FOCS*, pages 160–164, 1982.

Lattice-Based FHE as Secure as PKE

Zvika Brakerski*

Vinod Vaikuntanathan†

Abstract

We show that (leveled) fully homomorphic encryption (FHE) can be based on the hardness of $\tilde{O}(n^{1.5+\epsilon})$ -approximation for lattice problems (such as GapSVP) under quantum reductions for any $\epsilon > 0$ (or $\tilde{O}(n^{2+\epsilon})$ -approximation under classical reductions). This matches the best known hardness for “regular” (non-homomorphic) lattice based public-key encryption up to the ϵ factor. A number of previous methods had hit a roadblock at quasipolynomial approximation. (As usual, a circular security assumption can be used to achieve a non-leveled FHE scheme.)

Our approach consists of three main ideas: Noise-bounded sequential evaluation of high fan-in operations; Circuit sequentialization using Barrington’s Theorem; and finally, successive dimension-modulus reduction.

1 Introduction

Fully homomorphic encryption (FHE) allows us to convert an encryption of a message $\text{Enc}(m)$ into an encryption of a related message $\text{Enc}(f(m))$ for *any* efficient f , using only public information and without revealing anything about m itself. FHE has numerous theoretical and practical applications, the canonical one being to the problem of *outsourcing* computation to a remote server without compromising one’s privacy.

Until 2008, FHE was considered practically science fiction as no constructions or even viable approaches were known. A breakthrough by Gentry [Gen09b, Gen10, Gen09a] presented the first plausible candidate construction. The security of Gentry’s scheme relied on much stronger assumptions than standard (non homomorphic) public-key encryption (PKE), namely the hardness of problems on specially chosen ideal lattices as well as a new assumption called the sparse subset sum assumption. This state of affairs coincided with many researchers’ intuition that FHE, being much more versatile, should naturally be harder to achieve and should require stronger assumptions than regular public-key encryption. Brakerski and Vaikuntanathan [BV11] subsequently constructed an FHE scheme¹ based on the worst-case hardness of approximating lattice problems such as GapSVP (a promise version of the shortest vector problem on lattices) which have been studied extensively and are by now considered standard cryptographic assumptions. However, they required that the problem is hard to approximate to within a subexponential factor (in the dimension of the underlying lattice). This is in contrast to standard lattice-based public-key encryption

*Stanford University, zvika@stanford.edu. Supported by a Simons Postdoctoral Fellowship and by DARPA.

†MIT and University of Toronto, vinodv@mit.edu. Supported by an NSERC Discovery Grant, DARPA Grant number FA8750-11-2-0225, Connaught New Researcher Award and an Alfred P. Sloan Fellowship.

¹Here, and in the rest of the introduction, when we say FHE, we mean a leveled FHE scheme that can evaluate circuits of any a-priori bounded polynomial depth. The only known way to achieve non-leveled FHE schemes is to make a circular security assumption, in addition.

which can be based on the hardness of approximating the problem to within polynomial factors (explicitly $\tilde{O}(n^{1.5})$ using quantum reductions [Reg05] or $\tilde{O}(n^2)$ using classical reductions [Pei09]). Closing this gap has been a central goal in the study of FHE from both a theoretical and a practical perspective (since relying on a weaker assumption allows us to use shorter parameters resulting in better efficiency). Starting with [BGV12], several works using different approaches [Bra12, GSW13] have reduced the required factor of approximation to $n^{O(\log n)}$, which seemed to be a barrier for known methods.

In this work, we match the best known approximation factors up to any $\epsilon > 0$ and show that “science fiction” FHE can be as secure as any other lattice-based public-key encryption scheme. Furthermore, the keys and ciphertexts in our scheme (with the exception of the evaluation key which is only used for homomorphic evaluation) are identical to Regev’s original lattice-based PKE [Reg05], with parameters that are optimal up to a factor of $1 + \epsilon$.

Our results are summarized in the following theorem.

Theorem 1.1. *For every $\epsilon > 0$, there exists a leveled fully homomorphic encryption scheme based on the $\text{DLWE}_{n,q,\alpha}$ assumption (n -dimensional decisional LWE modulo q , with discrete Gaussian noise with parameter α), where $\alpha = 1/\tilde{O}(n^\epsilon \cdot \sqrt{n \log(q)})$.*

Thus, the scheme is secure based on either the quantum worst-case hardness of $\text{GapSVP}_{\tilde{O}(n^{1.5+\epsilon})}$, or the classical worst-case hardness of $\text{GapSVP}_{\tilde{O}(n^{2+\epsilon})}$.

High Level Overview. Our starting point is a new LWE-based FHE scheme by Gentry, Sahai and Waters [GSW13]. They present an encryption scheme where the public key is identical to Regev’s scheme, but the ciphertexts are square matrices rather than vectors. It was then possible to add and multiply ciphertexts using (roughly) matrix addition and multiplication. As in previous LWE-based FHE schemes, the ciphertext contains a “noise” element that grows with homomorphic operations and must be kept under a certain threshold in order for the ciphertext to be decryptable. The scheme is instantiated by a dimension n and modulus q , which correspond to the parameters of the LWE problem. The initial noise level is $\text{poly}(n)$ and the scheme is decryptable so long as the noise remains under (say) $q/8$. In order to base the scheme on the hardness of polynomial approximation to lattice problems, we would like to characterize the class of functions that can be homomorphically evaluated using $q = \text{poly}(n)$. The analysis of Gentry, Sahai and Waters [GSW13] shows that the evaluation of each Boolean gate increases the noise by a $\text{poly}(n)$ factor, and thus the class of functions that can be evaluated setting $q = \text{poly}(n)$ is NC^0 .

Our first observation is that the asymmetric (namely, non-commutative) nature of matrix multiplication gives rise to an interesting phenomenon in the GSW scheme: when multiplying two ciphertexts with noise levels e_1 and e_2 , the noise in the output turns out to be $e_1 + \text{poly}(n) \cdot e_2$. That is, the noise grows in an asymmetric manner. This means that if we want to multiply ℓ ciphertexts, for example, which all start with the same noise level, we can consecutively multiply them one after the other, and the final noise will only grow by a $\ell \cdot \text{poly}(n)$ factor. This is in contrast to the conventional wisdom that favors the use of a multiplication tree, which in this case would have resulted in a $\text{poly}(n)^{\log \ell}$ noise blowup. This observation already allows us to evaluate AC^0 circuits in a setting where the modulus $q = \text{poly}(n)$. (Using an additional trick, this can be extended to $\text{AC}^0[\oplus]$, namely AC^0 circuits augmented with XOR gates).

Our second idea is to push this technique forward by “sequentializing” larger circuit classes. A particularly potent tool in this direction of thought is Barrington’s Theorem [Bar89] which allows us to transform any NC^1 circuit into a polynomial length, width-5 permutation branching program.

Homomorphic evaluation of a length- ℓ branching program essentially requires homomorphically multiplying ℓ 5-by-5 encrypted permutation matrices, in contrast to the simple product operation on bits that we just accomplished. We show that this is in fact possible, namely a method of homomorphically multiplying ℓ permutation matrices that only increases the noise by an $\ell \cdot \text{poly}(n)$ factor. This gives us a way to evaluate any NC^1 circuit in a setting where the modulus $q = \text{poly}(n)$. In a high level, our technique here is reminiscent of Ishai and Paskin’s method of evaluating branching programs on encrypted data [IP07].

Evaluating NC^1 circuits with low noise blowup is a highly sought-after goal in the study of FHE schemes. The reason is Gentry’s bootstrapping theorem [Gen09b], which shows how to convert a scheme with some homomorphic properties into a fully homomorphic one, assuming that it can evaluate its own decryption circuit. Since the decryption circuit of the scheme in question lies in NC^1 , we can apply the bootstrapping theorem and obtain an FHE scheme with $q = \text{poly}(n)$, thus basing its security on the worst-case hardness of approximating lattice problems to within a (somewhat large) polynomial factor.

To obtain the optimal approximation factor (up to an arbitrarily small ϵ), we employ our third idea, namely a variant of the *dimension-modulus reduction* technique, originating in [BV11]. Our noise analysis of the NC^1 scheme shows that in order to obtain parameters that are optimal up to ϵ , the decryption circuit of our scheme must have depth at most $\epsilon \cdot \log(n)/2$, which seems unachievable. After all, an NC^1 circuit with n inputs and depth less than $\log n$ cannot even look at all the inputs! To solve this conundrum, we apply the *dimension-modulus reduction* technique, which allows us to “shrink” the ciphertext into a “smaller copy” of the same scheme. We show that by applying this method consecutively several times (as opposed to a single time as was done in [BV11]), we can reduce the ciphertext to a small enough size that decrypting it becomes possible in depth $\epsilon \log(n)/2$. This allows us to obtain an FHE scheme based on the worst-case hardness of approximating GapSVP within a factor of $\tilde{O}(n^{2+\epsilon})$ by classical algorithms, or a factor of $\tilde{O}(n^{1.5+\epsilon})$ by quantum algorithms.

Organization of the Paper. We start with some background and preliminaries: the reader should consult section 2.1 for background on Gaussian distributions, section 2.2 for the learning with error problem, and section 2.5 for homomorphic encryption. Our main result is described in Section 3 where we construct a (leveled) FHE scheme secure under the polynomial LWE assumption. We conclude in Section 4 by showing how to reduce and optimize the assumption to match the best known LWE assumption for lattice-based PKE.

2 Preliminaries

Matrices are denoted by bold-face capital letters, and vectors are denoted by bold-face small letters. All logarithms are taken to base 2, unless otherwise specified. For an integer q , we define the set $\mathbb{Z}_q \triangleq (-q/2, q/2] \cap \mathbb{Z}$. For any $x \in \mathbb{Q}$, we let $y = [x]_q$ denote the unique value $y \in (-q/2, q/2]$ such that $y \equiv x \pmod{q}$ (i.e. y is congruent to x modulo q).

We let κ denote a security parameter. When we speak of a negligible function $\text{negl}(\kappa)$, we mean a function that grows slower than $1/\kappa^c$ for any constant $c > 0$ and sufficiently large values of κ . When we say that an event happens with overwhelming probability, we mean that it happens with probability at least $1 - \text{negl}(\kappa)$ for some negligible function $\text{negl}(\kappa)$. We denote $y = \hat{O}_\kappa(x)$ if $y = O(x \cdot \text{polylog}(\kappa))$, and $y = \tilde{O}(x)$ if $y = \hat{O}_x(x)$. The notation $\tilde{\Theta}_\kappa(\cdot)$, $\tilde{\Omega}_\kappa(\cdot)$ is defined analogously.

The security parameter underlies all of our constructions. The parameters n, k etc. should all be considered to be a function of the security parameter κ , which is chosen according to the level of confidence desired by the user of the scheme. (The dimension of the LWE problem, defined below, should be considered to be polynomially related to the security parameter.)

2.1 Gaussians and Discrete Gaussians

In this work we will only consider one-dimensional Gaussians, and one-dimensional discrete Gaussians over the integers.

For $r > 0$, the (one-dimensional) Gaussian function $\rho_r : \mathbb{R} \rightarrow (0, 1]$ is defined as

$$\rho_r(x) \triangleq \exp(-\pi|x|^2/r^2).$$

The (spherical) continuous Gaussian distribution D_r is the distribution with density function proportional to ρ_r . The (one-dimensional, integer-coset) discrete Gaussian $D_{\mathbb{Z}-c, r}$ is the discrete distribution supported on $\mathbb{Z} - c$ for $c \in \mathbb{R}$, whose probability mass function is proportional to ρ_r .

Gaussian Rounding. To achieve the tightest results, we will need to use a simple Gaussian rounding procedure. The following is an immediate corollary of [BLP⁺13, Lemma 2.3].

Corollary 2.1. *There exists a randomized procedure $\lfloor \cdot \rfloor_G$ such that given $x \in \mathbb{R}$, it holds that $y \leftarrow \lfloor x \rfloor_G$ is such that $y - x \sim D_{\mathbb{Z}-x, 1}$.*

In fact, a slightly smaller standard deviation is achievable, but we use 1 for the sake of simplicity.

Sum of Discrete Gaussians. We wish to bound the absolute value of a sum of discrete Gaussians. The following are immediate corollaries from [Reg09, Corollary 3.10] and [GPV08, Lemma 3.1].

Proposition 2.2. *Let $\kappa \in \mathbb{N}$ be a security parameter. Then with all but $\text{negl}(\kappa)$ probability, if $x \sim D_r$, then $|x| \leq r \cdot \omega(\sqrt{\log \kappa})$. Similarly, if $x \sim D_{\mathbb{Z}-c, r}$ then with all but negligible probability, $|x| \leq \max\{r, \omega(\sqrt{\log \kappa})\} \cdot \omega(\sqrt{\log \kappa})$.*

Proposition 2.3. *Let $\kappa \in \mathbb{N}$ be a security parameter. Let $n \in \mathbb{N}$, let $\mathbf{z} \in \{0, 1\}^n$ and $\mathbf{c} \in \mathbb{R}^n$ be arbitrary, and let $\mathbf{e} \sim D_{\mathbb{Z}^n - \mathbf{c}, r}$. Then with all but negligible probability*

$$|\langle \mathbf{z}, \mathbf{e} \rangle| \leq \sqrt{n} \cdot \max\{r, \omega(\sqrt{\log \kappa})\} \cdot \omega(\sqrt{\log \kappa}) = \tilde{O}_\kappa(\sqrt{n}) \cdot r.$$

Proposition 2.4. *Let $\kappa \in \mathbb{N}$ be a security parameter. Let $n \in \mathbb{N}$, let $\mathbf{c} \in \mathbb{R}^n$ be arbitrary, let $\mathbf{e} \sim D_{\mathbb{Z}^n - \mathbf{c}, r}$, and let $\mathbf{z} \in \{0, 1\}^n$ be possibly dependent on \mathbf{e} . Then with all but negligible probability*

$$|\langle \mathbf{z}, \mathbf{e} \rangle| \leq n \cdot \max\{r, \omega(\sqrt{\log \kappa})\} \cdot \omega(\sqrt{\log \kappa}) = \tilde{O}_\kappa(n) \cdot r.$$

2.2 Learning With Errors (LWE)

The LWE problem was introduced by Regev [Reg05] as a generalization of “learning parity with noise”. For positive integers n and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^n$, and a probability distribution χ on \mathbb{Z} , let $A_{\mathbf{s}, \chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^n$ uniformly at random and a noise term $e \xleftarrow{\$} \chi$, and outputting $(\mathbf{a}, [\langle \mathbf{a}, \mathbf{s} \rangle + e]_q) \in \mathbb{Z}_q^n \times \mathbb{Z}_q$. Decisional LWE (DLWE) is defined as follows.

Definition 2.5 (DLWE). For an integer $q = q(n)$ and an error distribution $\chi = \chi(n)$ over \mathbb{Z} , the (average-case) decision learning with errors problem, denoted $\text{DLWE}_{n,m,q,\chi}$, is to distinguish (with non-negligible advantage) m samples chosen according to $A_{\mathbf{s},\chi}$ (for uniformly random $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^n \times \mathbb{Z}_q$. We denote by $\text{DLWE}_{n,q,\chi}$ the variant where the adversary gets oracle access to $A_{\mathbf{s},\chi}$, and is not a-priori bounded in the number of samples.

There are known quantum (Regev [Reg05]) and classical (Peikert [Pei09]) reductions between $\text{DLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be a discrete Gaussian distribution $D_{\mathbb{Z},\alpha q}$ for some $\alpha < 1$. We sometimes write $\text{DLWE}_{n,m,q,\alpha}$ (resp. $\text{DLWE}_{n,q,\alpha}$) to indicate this instantiation (it will be clear from the context when we use a distribution χ and when a Gaussian parameter α). We now state a corollary of the results of [Reg05, Pei09] (in conjunction with the search to decision reduction of Micciancio and Mol [MM11] and Micciancio and Peikert [MP11]). These results also extend to additional forms of q (see [MM11, MP11]).

Corollary 2.6 ([Reg05, Pei09, MM11, MP11]). Let $q = q(n) \in \mathbb{N}$ be either a prime power $q = p^r$, or a product of co-prime numbers $q = \prod q_i$ such that for all i , $q_i = \text{poly}(n)$, and let $\alpha \geq \sqrt{n}/q$. If there is an efficient algorithm that solves the (average-case) $\text{DLWE}_{n,q,\alpha}$ problem, then:

- There is an efficient quantum algorithm that solves $\text{GapSVP}_{\tilde{O}(n/\alpha)}$ (and $\text{SIVP}_{\tilde{O}(n/\alpha)}$) on any n -dimensional lattice.
- If in addition $q \geq \tilde{O}(2^{n/2})$, then there is an efficient classical algorithm for $\text{GapSVP}_{\tilde{O}(n/\alpha)}$ on any n -dimensional lattice.

Recall that GapSVP_γ is the (promise) problem of distinguishing, given a basis for a lattice and a parameter d , between the case where the lattice has a vector shorter than d , and the case where the lattice doesn't have any vector shorter than $\gamma \cdot d$. SIVP is the search problem of finding a set of “short” vectors. We refer the reader to [Reg05, Pei09] for more information.

The best known algorithms for GapSVP_γ ([Sch87]) require at least $2^{\tilde{\Omega}(n/\log \gamma)}$ time.

In this work, we will only consider the case where $q \leq 2^n$. Furthermore, the underlying security parameter κ is assumed to be polynomially related to the dimension n .

2.3 Vector Decomposition and Key Switching

We show how to decompose vectors in a way that makes their norm smaller, and yet preserves certain inner products. Our notation is generally adopted from [BGV12].

Vector Decomposition. We often break vectors into their bit representations as defined below:

- $\text{BitDecomp}_q(\mathbf{x})$: For $\mathbf{x} \in \mathbb{Z}^n$, let $w_{i,j} \in \{0, 1\}$ be such that $\mathbf{x}[i] = \sum_{j=0}^{\lceil \log q \rceil - 1} 2^j \cdot w_{i,j} \pmod{q}$. Output the vector

$$(w_{1,\lceil \log q \rceil - 1}, \dots, w_{1,0}, \dots, w_{n,\lceil \log q \rceil - 1}, \dots, w_{n,0}) \in \{0, 1\}^{n \cdot \lceil \log q \rceil}.$$

- $\text{PowersOfTwo}_q(\mathbf{y})$: For $\mathbf{y} \in \mathbb{Z}^n$, output

$$\left[(2^{\lceil \log q \rceil - 1} \mathbf{y}[1], \dots, 2\mathbf{y}[1], \mathbf{y}[1], \dots, 2^{\lceil \log q \rceil - 1} \cdot \mathbf{y}[n], \dots, 2\mathbf{y}[n], \mathbf{y}[n]) \right]_q \in \mathbb{Z}_q^{n \cdot \lceil \log q \rceil}.$$

We will usually omit the subscript q when it is clear from the context.

Claim 2.7. *For all $q \in \mathbb{N}$ and $\mathbf{x}, \mathbf{y} \in \mathbb{Z}^n$, it holds that*

$$\langle \mathbf{x}, \mathbf{y} \rangle = \langle \text{BitDecomp}_q(\mathbf{x}), \text{PowersOfTwo}_q(\mathbf{y}) \rangle \pmod{q}.$$

Additionally, we define the procedure **Flatten** following [GSW13], along with the procedure **Combine**. Let $\mathbf{g} = (2^{\lceil \log(q) \rceil - 1}, 2^{\lceil \log(q) \rceil - 2}, \dots, 4, 2, 1) \in \mathbb{Z}^{\lceil \log q \rceil}$ and let $\mathbf{G} := \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}^{n \times (n \cdot \lceil \log q \rceil)}$ denote the tensor product of \mathbf{g} with the n -by- n identity matrix \mathbf{I}_n .

- **Combine_q(z)**: For $\mathbf{z} \in \mathbb{Z}^{n \cdot \lceil \log q \rceil}$, output $[\mathbf{G} \cdot \mathbf{z}]_q \in \mathbb{Z}_q^n$.
- **Flatten_q(z)**: For $\mathbf{z} \in \mathbb{Z}^{n \cdot \lceil \log q \rceil}$, output $\text{BitDecomp}_q(\text{Combine}(\mathbf{z})) \in \{0, 1\}^{n \cdot \lceil \log q \rceil}$.

Claim 2.8. *For all $q \in \mathbb{N}$, and $\mathbf{x}, \mathbf{z} \in \mathbb{Z}^{n \cdot \lceil \log q \rceil}$, it holds that*

$$\langle \text{PowersOfTwo}(\mathbf{x}), \mathbf{z} \rangle = \langle \text{PowersOfTwo}(\mathbf{x}), \text{Flatten}(\mathbf{z}) \rangle \pmod{q}.$$

2.4 Partial Randomization Using LWE

We describe a procedure that allows us to partially randomize vectors while preserving their inner product with an LWE secret \mathbf{s} . This procedure will be useful to us when trying to manipulate ciphertexts that are a result of a homomorphic operation (and thus may have arbitrary dependence on the public parameters).

Let n, q, α be parameters for the DLWE problem, let $\chi = D_{\mathbb{Z}, \alpha q}$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ be some (arbitrary) vector.

- **RandParam(s)**: Let $m \triangleq (n + 1) \cdot (\log q + O(1))$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and $\mathbf{e} \xleftarrow{\$} \chi^m$. Compute $\mathbf{b} := [\mathbf{A} \cdot \mathbf{s} + \mathbf{e}]_q$, and define

$$\mathbf{P}_{\text{rand}} := [\mathbf{b} \parallel -\mathbf{A}] \in \mathbb{Z}_q^{m \times (n+1)}.$$

Output \mathbf{P}_{rand} .

We note that this is identical to the public key generation process in Regev's encryption scheme.

- **Rand(P_{rand}, c)**: For $\mathbf{c} \in \mathbb{Z}_q^{n+1}$, sample $\mathbf{r} \xleftarrow{\$} \{0, 1\}^m$, and compute

$$\mathbf{c}_{\text{rand}} := [\mathbf{c} + \mathbf{r}^T \mathbf{P}_{\text{rand}}]_q.$$

Output \mathbf{c}_{rand} .

The properties of this process are summarized below.

First, we state the security of our procedure, namely that \mathbf{P}_{rand} does not reveal information about \mathbf{s} . The proof is straightforward and omitted.

Lemma 2.9. *If \mathbf{s} is uniformly sampled and $\mathbf{P}_{\text{rand}} \leftarrow \text{RandParam}(\mathbf{s})$, then under the $\text{DLWE}_{n,q,\alpha}$ assumption, \mathbf{P}_{rand} is computationally indistinguishable from uniform.*

Next, we state that the inner product of the randomized vector with $(1, \mathbf{s})$ does not change by much.

Lemma 2.10. Let $\mathbf{s} \in \mathbb{Z}_q^n$ be arbitrary, and let $\mathbf{P}_{\text{rand}} \leftarrow \text{RandParam}(\mathbf{s})$. Let $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ be arbitrary, and $\mathbf{c}_{\text{rand}} \leftarrow \text{Rand}(\mathbf{P}_{\text{rand}}, \mathbf{c})$, then there exists δ such that

$$\langle \mathbf{c}, (1, \mathbf{s}) \rangle - \langle \mathbf{c}_{\text{rand}}, (1, \mathbf{s}) \rangle = \delta \pmod{q},$$

and $|\delta| \leq \tilde{O}_\kappa(\sqrt{n \log(q)}) \cdot \alpha q$ with all but $\text{negl}(\kappa)$ probability.

Proof. We start by noting that

$$\langle \mathbf{r}^T \mathbf{P}_{\text{rand}}, (1, \mathbf{s}) \rangle = \langle \mathbf{r}, \mathbf{e} \rangle \pmod{q},$$

where \mathbf{e} is the noise used to generate \mathbf{P}_{rand} . Using Proposition 2.3, the result follows. \square

Finally, we state the randomization property of our procedure.

Lemma 2.11. Let $q \leq 2^n$. Let $\mathbf{s} \in \mathbb{Z}_q^n$ be arbitrary, and let $\mathbf{P}_{\text{rand}} \leftarrow \text{RandParam}(\mathbf{s})$. Let $\mathbf{f} \xleftarrow{\$} D_{\mathbb{Z}, t}^{(n+1) \cdot \lceil \log(q) \rceil}$ for some t , and let $\mathbf{c} \in \mathbb{Z}_q^{n+1}$ be arbitrary (possibly dependent on \mathbf{f}). Finally, let $\mathbf{c}_{\text{rand}} \leftarrow \text{Rand}(\mathbf{P}_{\text{rand}}, \mathbf{c})$, then

$$|\langle \text{BitDecomp}_q(\mathbf{c}_{\text{rand}}), \mathbf{f} \rangle| \leq \tilde{O}_\kappa(\sqrt{n \log(q)}) \cdot t,$$

with all but $\text{negl}(\kappa)$ probability.

Proof. By the leftover hash lemma, the last n coordinates of \mathbf{c}_{rand} are distributed uniformly, and independently of \mathbf{f}, \mathbf{c} . By Proposition 2.3, this part of \mathbf{c}_{rand} contributes $\tilde{O}_\kappa(\sqrt{n \log(q)}) \cdot t$ to the inner product (with all but negligible probability).

The first coordinate of \mathbf{c}_{rand} may have dependence on \mathbf{f} , but it only decomposes to $O(\log q)$ bits, and therefore by Proposition 2.4, its contribution to the inner product is at most $\tilde{O}_\kappa(\log(q)) \cdot t$ with all but negligible probability. Recalling that $q \leq 2^n$, this is at most $\tilde{O}_\kappa(\sqrt{n \log(q)}) \cdot t$.

The union bound completes the proof. \square

2.5 Homomorphic Encryption and Bootstrapping

We now define homomorphic encryption and introduce Gentry's bootstrapping theorem. Our definitions are mostly taken from [BV11, BGV12].

A homomorphic (public-key) encryption scheme $\text{HE} = (\text{HE.Keygen}, \text{HE.Enc}, \text{HE.Dec}, \text{HE.Eval})$ is a quadruple of PPT algorithms as follows (κ is the security parameter):

- **Key generation** $(pk, evk, sk) \leftarrow \text{HE.Keygen}(1^\kappa)$: Outputs a public encryption key pk , a public evaluation key evk and a secret decryption key sk .²
- **Encryption** $c \leftarrow \text{HE.Enc}_{pk}(\mu)$: Using the public key pk , encrypts a single bit message $\mu \in \{0, 1\}$ into a ciphertext c .
- **Decryption** $\mu \leftarrow \text{HE.Dec}_{sk}(c)$: Using the secret key sk , decrypts a ciphertext c to recover the message $\mu \in \{0, 1\}$.
- **Homomorphic evaluation** $c_f \leftarrow \text{HE.Eval}_{evk}(f, c_1, \dots, c_\ell)$: Using the evaluation key evk , applies a function $f : \{0, 1\}^\ell \rightarrow \{0, 1\}$ to c_1, \dots, c_ℓ , and outputs a ciphertext c_f .

²We adopt the terminology of [BV11] that treats the evaluation key as a separate entity from the public key.

A homomorphic encryption scheme is said to be secure if it is semantically secure (note that the adversary is given both pk and evk).

Homomorphism w.r.t depth-bounded circuits and full homomorphism are defined next:

Definition 2.12 (compactness and full homomorphism). *A homomorphic encryption scheme is compact if its decryption circuit is independent of the evaluated function. A compact scheme is (pure) fully homomorphic if it can evaluate any efficiently computable function. The scheme is leveled fully homomorphic if it takes 1^L as additional input in key generation, and can only evaluate depth L Boolean circuits.*

Gentry’s bootstrapping theorem shows how to go from limited amount of homomorphism to full homomorphism. This method has to do with the *augmented decryption circuit*.

Definition 2.13. *Consider a homomorphic encryption scheme HE. Let (sk, pk, evk) be properly generated keys and let \mathcal{C} be the set of properly decryptable ciphertexts. Then the set of augmented decryption functions, $\{f_{c_1, c_2}\}_{c_1, c_2 \in \mathcal{C}}$ is defined by $f_{c_1, c_2}(x) = \text{HE.Dec}_x(c_1) \wedge \text{HE.Dec}_x(c_2)$. Namely, the function that uses its input as secret key, decrypts c_1, c_2 and returns the NAND of the results.*

The bootstrapping theorem is thus as follows.

Theorem 2.14 (bootstrapping [Gen09b, Gen09a]). *A scheme that can homomorphically evaluate its family of augmented decryption circuits can be transformed into a leveled fully homomorphic encryption scheme with the same decryption circuit, ciphertext space and public key.*

Furthermore, if the aforementioned scheme is also weak circular secure (remains secure even against an adversary who gets encryptions of the bits of the secret key), then it can be made into a pure fully homomorphic encryption scheme.

3 Our FHE Scheme

In this section, we describe an FHE scheme secure under a polynomial LWE assumption which, using known reductions [Reg05, Pei09], translates to the worst-case hardness of solving various lattice problems to within polynomial approximation factors. We start with the basic encryption scheme in Section 3.1, and describe “proto-homomorphic” addition and multiplication subroutines in Section 3.2. Departing from the “conventional wisdom” in FHE, our circuit evaluation procedure in Section 3.3 will *not* be a naive combination of these proto-homomorphic operations, but rather a carefully designed procedure that manages the noise growth effectively.

Finally, in Section 3.4, we put this all together to get our FHE scheme under the decisional LWE assumption $\text{DLWE}_{n, q, \alpha}$ with $\alpha = n^{-c}$ for some constant $c > 0$. This polynomial factor is rather large: thus, in Section 4, we apply a carefully designed variant of the dimension-modulus reduction procedure of [BV11] to obtain our final FHE scheme that is secure under the hardness of $\text{DLWE}_{n, q, \alpha}$ with $\alpha \leq 1/\tilde{O}_\kappa(n^\epsilon \cdot \sqrt{n \log(q)})$ which is weakest LWE hardness assumption that underlies the (non-homomorphic) lattice-based PKE schemes [AD97, Reg04, Reg05, Pei09, BLP⁺13].

3.1 The Basic Encryption Scheme

Our basic encryption scheme closely follows the Gentry-Sahai-Waters FHE scheme [GSW13]. We refer the reader to Section 2.3 for the description of the vector decomposition routines `PowersOfTwo`, `BitDecomp` and `Flatten` used in the scheme below.

System Parameters. Let n be the LWE dimension and q be an LWE modulus. Define $N := (n+1) \cdot \lceil \log q \rceil$. Let d denote the maximal homomorphism depth that is allowed by the scheme. Let χ be an error distribution over \mathbb{Z} . Typically χ will be a discrete Gaussian $D_{\mathbb{Z}, \alpha q}$ for $\alpha = 1/\tilde{O}_\kappa(\sqrt{n \log(q)} \cdot 4^d)$. Recall that we identify \mathbb{Z}_q with the set $(-q/2, q/2] \cap \mathbb{Z}$.

- **NCCrypt.Keygen**($1^n, q, 4^d$): Sample a vector $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$. Let $m \triangleq (n+1) \cdot (\log q + O(1))$. Sample $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{m \times n}$ and $\mathbf{e} \xleftarrow{\$} \chi^m$. Compute $\mathbf{b} := [\mathbf{A} \cdot \mathbf{s} + \mathbf{e}]_q$, and define

$$\mathbf{P} := [\mathbf{b} \parallel -\mathbf{A}] \in \mathbb{Z}_q^{m \times (n+1)}.$$

Output $sk = \mathbf{s}$ and $pk = evk = \mathbf{P}$.

We describe public-key as well as secret-key encryption algorithms. Looking ahead, we remark that a secret-key encryption of μ is somewhat less “noisy” than a public-key encryption of μ .

- **NCCrypt.PubEnc**(pk, μ): To encrypt a bit $\mu \in \{0, 1\}$, using the public key $pk = \mathbf{P}$, we let $\mathbf{R} \xleftarrow{\$} \{0, 1\}^{N \times m}$, and output the ciphertext

$$\mathbf{C} = \text{Flatten}(\text{BitDecomp}(\mathbf{R} \cdot \mathbf{P}) + \mu \cdot \mathbf{I}) \in \{0, 1\}^{N \times N}.$$

- **NCCrypt.SecEnc**(sk, μ): A symmetric encryption of a bit $\mu \in \{0, 1\}$, using the secret key $sk = \mathbf{s}$, is performed by sampling $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{N \times n}$ and $\mathbf{e} \xleftarrow{\$} \chi^N$, computing $\mathbf{b} := [\mathbf{A} \cdot \mathbf{s} + \mathbf{e}]_q$, and defining

$$\mathbf{C} = \text{Flatten}(\text{BitDecomp}([\mathbf{b} \parallel -\mathbf{A}]) + \mu \cdot \mathbf{I}) \in \{0, 1\}^{N \times N}.$$

- **NCCrypt.Dec**(sk, \mathbf{C}): Let \mathbf{c} be the second row of \mathbf{C} . We use standard Regev decryption on \mathbf{c} . Namely, we output $\mu^* = 0$ if $\left| \langle \mathbf{c}, \text{PowersOfTwo}(1, \mathbf{s}) \rangle \right|_q < q/8$, and $\mu^* = 1$ otherwise.

Correctness. In order to show correctness of this scheme, we analyze the noise magnitude of ciphertexts produced by both the public-key and secret-key encryption algorithms. As we will show shortly, for ciphertexts \mathbf{C} produced by either encryption algorithm, we have

$$\mathbf{C} \cdot \text{PowersOfTwo}(1, \mathbf{s}) = \mu \cdot \text{PowersOfTwo}(1, \mathbf{s}) + \mathbf{e} \pmod{q}$$

for a noise vector \mathbf{e} of “small magnitude”. This motivates our definition of the noise in the ciphertext \mathbf{C} with respect to a secret key vector \mathbf{s} and a message μ as follows.

Definition 3.1. For every $\mathbf{C} \in \{0, 1\}^{N \times N}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mu \in \mathbb{Z}$, we define

$$\text{noise}_{\mathbf{s}, \mu}(\mathbf{C}) \triangleq \|(\mathbf{C} - \mu \mathbf{I}) \cdot \text{PowersOfTwo}(1, \mathbf{s}) \pmod{q}\|_\infty$$

The significance of this definition is captured by the following claims. The first claim shows that any ciphertext with small noise is decrypted correctly.

Lemma 3.2. For every $\mathbf{C} \in \{0, 1\}^{N \times N}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mu \in \mathbb{Z}$ such that $\text{noise}_{\mathbf{s}, \mu}(\mathbf{C}) < q/8$,

$$\text{NCCrypt.Dec}(\mathbf{s}, \mathbf{C}) = \mu$$

Proof. Since $\text{noise}_{\mathbf{s},\mu}(\mathbf{C}) < q/8$, we have

$$\mathbf{C} \cdot \text{PowersOfTwo}(1, \mathbf{s}) = \boldsymbol{\eta} + \mu \cdot \text{PowersOfTwo}(1, \mathbf{s}) \pmod{q}$$

where $\|\boldsymbol{\eta}\|_\infty < q/8$. Thus, for the second row of \mathbf{C} it holds that

$$\langle \mathbf{c}, \text{PowersOfTwo}(1, \mathbf{s}) \rangle = \eta + 2^{\lceil \log(q) \rceil - 2} \cdot \mu \pmod{q}$$

where $|\eta| < q/8$. Thus, when $\mu = 0$,

$$\left| \langle \mathbf{c}, \text{PowersOfTwo}(1, \mathbf{s}) \rangle \right|_q = |\eta| < q/8$$

When $\mu = 1$,

$$\left| \langle \mathbf{c}, \text{PowersOfTwo}(1, \mathbf{s}) \rangle \right|_q \geq q/4 - |\eta| \geq q/8$$

since $q/4 \leq 2^{\lceil \log(q) \rceil - 2} < q/2$. This shows correctness of decryption for ciphertexts with small noise. \square

The next claim demonstrates parameter settings for which the (public key and secret key) encryption algorithms produce ciphertexts with small noise.

Lemma 3.3. *Let n be the LWE dimension, q be the LWE modulus and $\chi = D_{\mathbb{Z}, \alpha q}$ be the discrete Gaussian distribution. Then, for every $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mu \in \{0, 1\}$,*

- for $\mathbf{C}_{\text{pub}} \leftarrow \text{NCCrypt.PubEnc}(pk, \mu)$, we have $\text{noise}_{\mathbf{s},\mu}(\mathbf{C}_{\text{pub}}) = \tilde{O}_\kappa(\alpha q \cdot \sqrt{m})$.
- for $\mathbf{C}_{\text{sec}} \leftarrow \text{NCCrypt.SecEnc}(sk, \mu)$, we have $\text{noise}_{\mathbf{s},\mu}(\mathbf{C}_{\text{sec}}) = \tilde{O}_\kappa(\alpha q)$.

with all but negligible (in κ) probability over the coins of NCCrypt.Keygen .

In particular, we have correctness of decryption for the public key encryption for $\alpha < 1/\tilde{\Omega}_\kappa(\sqrt{m})$, and for the secret key encryption for $\alpha < 1/\Omega_\kappa(1)$.

Proof. We first show the analysis for the public-key encryption.

$$\begin{aligned} \mathbf{C}_{\text{pub}} \cdot \text{PowersOfTwo}(1, \mathbf{s}) &= \text{Flatten} \left(\text{BitDecomp}(\mathbf{R} \cdot \mathbf{P}) + \mu \cdot \mathbf{I} \right) \cdot \text{PowersOfTwo}(1, \mathbf{s}) \\ &= \left(\text{BitDecomp}(\mathbf{R} \cdot \mathbf{P}) + \mu \cdot \mathbf{I} \right) \cdot \text{PowersOfTwo}(1, \mathbf{s}) \\ &= \mathbf{R} \cdot \mathbf{P} \cdot (1, \mathbf{s})^T + \mu \cdot \text{PowersOfTwo}(1, \mathbf{s}) \\ &= \mathbf{R} \cdot \mathbf{e} + \mu \cdot \text{PowersOfTwo}(1, \mathbf{s}) \end{aligned}$$

Thus, by Proposition 2.3,

$$\text{noise}(\mathbf{C}_{\text{pub}}) = \|\mathbf{R} \cdot \mathbf{e}\|_\infty = \tilde{O}_\kappa(\sqrt{m} \cdot \alpha q)$$

This is less than $q/8$ by the choice of $\alpha < 1/\tilde{\Omega}_\kappa(\sqrt{m})$.

The analysis for the secret key encryption follows analogously, except that

$$\mathbf{C}_{\text{sec}} \cdot \text{PowersOfTwo}(1, \mathbf{s}) = \mathbf{e} + \mu \cdot \text{PowersOfTwo}(1, \mathbf{s})$$

Thus,

$$\text{noise}(\mathbf{C}_{\text{sec}}) = \|\mathbf{e}\|_\infty = \tilde{O}_\kappa(\alpha q)$$

which is less than $q/8$ by the choice of $\alpha < 1/\tilde{\Omega}_\kappa(1)$. \square

Security. Semantic security of the scheme follows from the decisional LWE assumption $\text{DLWE}_{n,q,\alpha}$, similarly to Regev’s encryption scheme (see [Reg05, BV11, GSW13] for similar arguments).

Complexity of Decryption. Our decryption algorithm is essentially the same as the decryption algorithm in Regev’s encryption scheme, the complexity of which has been thoroughly studied in the context of FHE. The following is an immediate corollary from [BV11, Lemma 4.1].

Proposition 3.4. *There exists a constant c_{dec} such that the decryption circuit of the scheme NCCrypt , with parameters n, q , has depth at most $c_{\text{dec}} \cdot \log(n \log q)$.*

3.2 Proto-Homomorphic Operations

We now describe proto-homomorphic addition and multiplication algorithms which will be used in Section 3.3 for homomorphic circuit evaluation. Departing from the “conventional wisdom” in FHE, our circuit evaluation procedure will *not* be a naive combination of homomorphic addition and multiplication, but a carefully designed procedure that manages the noise growth effectively. To further stress the fact that we do not intend for these procedures to be used independently, we call them proto-homomorphic operations.

Proto-Homomorphic Addition. This is a simple addition of the ciphertext matrices.

- $\text{NCCrypt.ProtoAdd}(\mathbf{C}_1, \mathbf{C}_2)$: Output $\mathbf{C}_+ := \text{Flatten}(\mathbf{C}_1 + \mathbf{C}_2)$.

Jumping ahead, we note that in our use of NCCrypt.ProtoAdd in Section 3.3, both \mathbf{C}_1 and \mathbf{C}_2 will be encryptions of bits, and at most one of them will be an encryption of 1. The following claim analyzes the noise growth in homomorphic addition.

Claim 3.4.1 (Noise Growth in NCCrypt.ProtoAdd). *For every $\mathbf{s} \in \mathbb{Z}_q^n$, $\mu_1, \mu_2 \in \mathbb{Z}$ and $\mathbf{C}_1, \mathbf{C}_2 \in \{0, 1\}^{N \times N}$, we have*

$$\text{noise}_{\mathbf{s}, \mu_1 + \mu_2}(\text{NCCrypt.ProtoAdd}(\mathbf{C}_1, \mathbf{C}_2)) \leq \text{noise}_{\mathbf{s}, \mu_1}(\mathbf{C}_1) + \text{noise}_{\mathbf{s}, \mu_2}(\mathbf{C}_2)$$

Proof. Let $\mathbf{C}_+ \leftarrow \text{NCCrypt.ProtoAdd}(\mathbf{C}_1, \mathbf{C}_2)$. We note that

$$\begin{aligned} \mathbf{C}_+ \cdot \text{PowersOfTwo}(1, \mathbf{s}) &= \text{Flatten}(\mathbf{C}_1 + \mathbf{C}_2) \cdot \text{PowersOfTwo}(1, \mathbf{s}) \\ &= (\mathbf{C}_1 + \mathbf{C}_2) \cdot \text{PowersOfTwo}(1, \mathbf{s}) \\ &= \mathbf{C}_1 \cdot \text{PowersOfTwo}(1, \mathbf{s}) + \mathbf{C}_2 \cdot \text{PowersOfTwo}(1, \mathbf{s}) \\ &= (\mathbf{e}_1 + \mathbf{e}_2) + (\mu_1 + \mu_2) \cdot \mathbf{I} \end{aligned}$$

Thus, by the definition of $\text{noise}_{\mathbf{s}, \mu_1 + \mu_2}$, we have

$$\text{noise}_{\mathbf{s}, \mu_1 + \mu_2}(\mathbf{C}_+) \leq \text{noise}_{\mathbf{s}, \mu_1}(\mathbf{C}_1) + \text{noise}_{\mathbf{s}, \mu_2}(\mathbf{C}_2)$$

□

Homomorphic Multiplication. This is essentially a multiplication of the ciphertext matrices, except that we randomize the first ciphertext.

- $\text{NCCrypt.ProtoMult}(evk, \mathbf{C}_1, \mathbf{C}_2)$:

- Randomize $\mathbf{C}_1 \in \{0, 1\}^{N \times N}$ into a matrix $\tilde{\mathbf{C}}_1 \in \{0, 1\}^{N \times N}$ by replacing each row \mathbf{c} in \mathbf{C}_1 by the row

$$\tilde{\mathbf{c}} \leftarrow \text{BitDecomp}(\text{Rand}(pk, \text{Combine}(\mathbf{c})))$$

where Rand is the LWE randomization procedure from Section 2.4.

- Output $\mathbf{C}_\times \leftarrow \text{Flatten}(\tilde{\mathbf{C}}_1 \cdot \mathbf{C}_2)$.

Jumping ahead, we remark that when we use NCCrypt.ProtoMult in our homomorphic circuit evaluation in Section 3.3, the first ciphertext will be an “evaluated ciphertext” (namely, a result of previous homomorphic evaluations), whereas the second ciphertext will be a “fresh ciphertext” (namely an output of the *secret key* encryption algorithm).

The first new idea in this work is that while the order of the arguments does not matter in homomorphic addition, the homomorphic multiplication algorithm NCCrypt.ProtoMult is inherently asymmetric, since it is essentially the (non-commutative) matrix multiplication operation. This asymmetry turns out to be the key to achieving improved noise growth, as Claim 3.4.2 below will demonstrate.

Claim 3.4.2 (Noise Growth in NCCrypt.ProtoMult). *For every $\mathbf{s} \in \mathbb{Z}_q^n$, $\mu_1, \mu_2 \in \{0, 1\}$ and $\mathbf{C}_1 \in \{0, 1\}^{N \times N}$ and $\mathbf{C}_2 \leftarrow \text{NCCrypt.SecEnc}(sk, \mu_2)$, we have*

$$\text{noise}_{\mathbf{s}, \mu_1 \mu_2}(\text{NCCrypt.ProtoMult}(\mathbf{C}_1, \mathbf{C}_2)) \leq |\mu_2| \cdot \text{noise}_{\mathbf{s}, \mu_1}(\mathbf{C}_1) + \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q})$$

with all but negligible probability over the randomness of NCCrypt.Keygen , NCCrypt.SecEnc and NCCrypt.ProtoMult .

Remark. In words, Claim 3.4.2 says that if $\mu_2 \in \{0, 1\}$ (as will be the case in our homomorphic circuit evaluation in Section 3.3), the noise in \mathbf{C}_\times is at most the noise in \mathbf{C}_1 , plus a fixed additive term. What’s more, if $\mu_2 = 0$, then the noise in \mathbf{C}_\times is independent of that in \mathbf{C}_1 ! *These two facts are the key new ideas that enable our main result.*

Proof. (of Claim 3.4.2.) Let $\mathbf{C}_\times \leftarrow \text{NCCrypt.ProtoMult}(evk, \mathbf{C}_1, \mathbf{C}_2)$. Note that

$$\begin{aligned} \mathbf{C}_\times \cdot \text{PowersOfTwo}(1, \mathbf{s}) &= \text{Flatten}(\tilde{\mathbf{C}}_1 \cdot \mathbf{C}_2) \cdot \text{PowersOfTwo}(1, \mathbf{s}) \\ &= \tilde{\mathbf{C}}_1 \cdot (\mathbf{C}_2 \cdot \text{PowersOfTwo}(1, \mathbf{s})) \\ &= \tilde{\mathbf{C}}_1 \cdot (\mathbf{e}_2 + \mu_2 \cdot \text{PowersOfTwo}(1, \mathbf{s})) \\ &= \tilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 + \mu_2 \cdot \tilde{\mathbf{C}}_1 \cdot \text{PowersOfTwo}(1, \mathbf{s}) \\ &= \tilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 + \mu_2 \cdot (\tilde{\mathbf{e}}_1 + \mu_1 \cdot \text{PowersOfTwo}(1, \mathbf{s})) \\ &= (\tilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 + \mu_2 \cdot \tilde{\mathbf{e}}_1) + \mu_1 \mu_2 \cdot \text{PowersOfTwo}(1, \mathbf{s}) \end{aligned} \tag{1}$$

Since $\mathbf{e}_2 \leftarrow D_{\mathbb{Z}, \alpha q}^N$ and each row of $\tilde{\mathbf{C}}_1 \in \{0, 1\}^{N \times N}$ is the result of Rand , by Lemma 2.11, we have

$$\left\| \tilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 \right\|_\infty \leq \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) \tag{2}$$

with all but negligible probability.
Also, by lemma 2.10, we have

$$\|\tilde{\mathbf{e}}_1\|_\infty \leq \|\mathbf{e}_1\|_\infty + \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) \quad (3)$$

with all but negligible probability.

Putting together Eq. (1),(2) and (3), we have

$$\text{noise}_{\mathbf{s}, \mu_2}(\mathbf{C}_\times) \leq \left\| \tilde{\mathbf{C}}_1 \cdot \mathbf{e}_2 \right\|_\infty + |\mu_2| \cdot \|\tilde{\mathbf{e}}_1\|_\infty \leq |\mu_2| \cdot \text{noise}_{\mathbf{s}, \mu_1}(\mathbf{C}_1) + \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q})$$

which finishes the proof. \square

3.3 Homomorphic Evaluation of Circuits

We now describe how to homomorphically evaluate a Boolean circuit Ψ with two-input NAND gates that takes ℓ inputs, and has depth d . In particular, our scheme will be able to evaluate circuits of depth $c \cdot \log n$ (for any constant c) under a polynomial LWE assumption, namely $\text{DLWE}_{n,q,\chi}$ where $\chi = D_{\mathbb{Z},\alpha}$ and $\alpha = 1/n^{\Theta(c)}$. Since the depth of the decryption circuit is $c_{\text{dec}} \cdot \log(n \log(q)) \leq 2c_{\text{dec}} \cdot \log(n)$ (for some constant $c_{\text{dec}} > 0$), the scheme is bootstrappable, and by the bootstrapping theorem (Theorem 2.14), we get a leveled FHE scheme under the same assumption.

To evaluate a circuit, our scheme first turns it into a width-5 permutation branching program [BDFP86, Bar89], a model of computation that we describe below.

Width-5 Permutation Branching Programs. A permutation branching program Π of length L with input space $\{0,1\}^\ell$ is a sequence of L tuples of the form $(\text{var}(t), \sigma_{t,0}, \sigma_{t,1})$ where

- $\text{var} : [L] \rightarrow [\ell]$ is a function that associates the t -th tuple with an input bit $x_{\text{var}(t)}$.
- $\sigma_{j,0}$ and $\sigma_{j,1}$ are permutations on 5 elements. We will think of $\sigma_{j,0}$ and $\sigma_{j,1}$ as bijective functions from the set $\{1, 2, 3, 4, 5\}$ to itself.

The computation of the program Π on input $\mathbf{x} = (x_1, \dots, x_\ell)$ proceeds as follows. The state of the computation at any point in time t is a number $\zeta_t \in \{1, 2, 3, 4, 5\}$. Computation starts with the initial state $\zeta_0 = 1$. The state ζ_t is computed recursively as

$$\zeta_t = \sigma_{t, \text{var}(t)}(\zeta_{t-1})$$

Finally, after L steps, our state is ζ_L . The output of the computation is 1 if $\zeta_L = 1$, and 0 otherwise.

To manage the growth of noise in homomorphic evaluation, we need to work with bits rather than numbers. Thus, we prefer to represent the state $\zeta_t \in \{1, 2, 3, 4, 5\}$ by a 0-1 vector \mathbf{v}_t which is the unit vector \mathbf{u}_{ζ_t} in 5 dimensions.

The computation then proceeds as follows. The idea is that $\mathbf{v}_t[i] = 1$ if and only if $\sigma_{t, \text{var}(t)}(\zeta_{t-1}) = i$. Turning this around, $\mathbf{v}_t[i] = 1$ if and only if either:

- $\mathbf{v}_{t-1}[\sigma_{t,0}^{-1}(i)] = 1$ and $x_{\text{var}(t)} = 0$; or
- $\mathbf{v}_{t-1}[\sigma_{t,1}^{-1}(i)] = 1$ and $x_{\text{var}(t)} = 1$.

The following formula captures this condition. For $t = 1, \dots, L$, and $i \in \{1, 2, 3, 4, 5\}$, we have:

$$\begin{aligned}\mathbf{v}_t[i] &:= \mathbf{v}_{t-1}[\sigma_{t,0}^{-1}(i)] \cdot (1 - x_{\text{var}(t)}) + \mathbf{v}_{t-1}[\sigma_{t,1}^{-1}(i)] \cdot x_{\text{var}(t)} \\ &= \mathbf{v}_{t-1}[\gamma_{t,i,0}] \cdot (1 - x_{\text{var}(t)}) + \mathbf{v}_{t-1}[\gamma_{t,i,1}] \cdot x_{\text{var}(t)}\end{aligned}\tag{4}$$

where $\gamma_{t,i,0} \triangleq \sigma_{t,0}^{-1}(i)$ and $\gamma_{t,i,1} \triangleq \sigma_{t,1}^{-1}(i)$ are constants that are publicly computable given the description of the branching program. It is this form that we will work with in our homomorphic evaluation.

The important property that we will use is that circuits of depth d can be simulated by branching programs of depth $L = 4^d$.

Theorem 3.5 (Barrington's Theorem [Bar89]). *Every Boolean NAND circuit Ψ that acts on ℓ inputs and has depth d can be computed by a width-5 permutation branching program Π of length 4^d . Given the description of the circuit Ψ , the description of the branching program Π can be computed in $\text{poly}(\ell, 4^d)$ time.*

Homomorphic Evaluation $\text{NCCrypt.Eval}(\Psi, \mathbf{C}_1, \dots, \mathbf{C}_\ell)$. The homomorphic evaluation procedure will first convert the depth- d circuit Ψ into a width-5 permutation branching program Π of length $L = 4^d$.

- **[Initialization]** We will maintain the encrypted state of the computation of the branching program for every step t . We denote this by $\mathbf{V}_t = (\mathbf{V}_{t,1}, \mathbf{V}_{t,2}, \mathbf{V}_{t,3}, \mathbf{V}_{t,4}, \mathbf{V}_{t,5})$, where each $\mathbf{V}_t[i] \in \{0, 1\}^{N \times N}$ will be an encryption of $\mathbf{v}_t[i]$.
 - We initialize the state as follows. Compute $\mathbf{V}_{0,i} := \mathbf{v}_0[i] \cdot \mathbf{I}$.
Note that $\mathbf{V}_{0,i}$ is in fact a valid encryption of the bit $\mathbf{v}_0[i]$ with zero noise.
 - We also compute encryptions of the complements of the input bits, for convenience. That is, set $\bar{\mathbf{C}}_k := \mathbf{I} - \mathbf{C}_k$. Note that $\bar{\mathbf{C}}_k$ is an encryption of $\bar{x}_k = 1 - x_k$ with the same noise as \mathbf{C}_k .
- **[Evaluation]** The evaluation proceeds iteratively for $t = 1, \dots, L$, where L is the length of the branching program Π . Assuming that we have $\mathbf{V}_{t-1} := (\mathbf{V}_{t-1,1}, \mathbf{V}_{t-1,2}, \dots, \mathbf{V}_{t-1,5})$, the encryption of the state of the branching program computation at time $t - 1$, we compute $\mathbf{V}_t := (\mathbf{V}_{t,1}, \mathbf{V}_{t,2}, \dots, \mathbf{V}_{t,5})$ by homomorphically evaluating Eq. (4) above.

That is, for $i \in \{1, 2, 3, 4, 5\}$, we compute

$$\mathbf{V}_{t,i} := \text{NCCrypt.ProtoAdd}\left(\text{NCCrypt.ProtoMult}(\mathbf{V}_{t-1,\gamma_0}, \bar{\mathbf{C}}_{\text{var}(t)}),\tag{5}$$

$$\text{NCCrypt.ProtoMult}(\mathbf{V}_{t-1,\gamma_1}, \mathbf{C}_{\text{var}(t)})\right)\tag{6}$$

- **[Output]** Upon finishing the evaluation stage, we have $\mathbf{V}_L := (\mathbf{V}_{L,1}, \mathbf{V}_{L,2}, \dots, \mathbf{V}_{L,5})$. Output $\mathbf{V}_{L,1}$ as the result of the homomorphic evaluation.

We now show that the scheme correctly evaluates circuits.

Lemma 3.6 (Correctness of Homomorphic Evaluation). *Let n be the LWE dimension, q the LWE modulus, Ψ be any Boolean circuit of depth d , and*

$$\alpha \leq 1/\tilde{\Theta}_\kappa(4^d \cdot \sqrt{n \log q})$$

For every $x_1, \dots, x_\ell \in \{0, 1\}$, every Boolean circuit Ψ of depth at most d , and every secret key sk , letting $\mathbf{C}_k \leftarrow \text{NCCrypt.SecEnc}(sk, x_k)$ be the secret key encryptions of the inputs, and $\mathbf{C}_\Psi \leftarrow \text{NCCrypt.Eval}(evk, \Psi, \mathbf{C}_1, \dots, \mathbf{C}_k)$ be the evaluated ciphertext, we have:

$$\text{NCCrypt.Dec}(sk, \mathbf{C}_\Psi) = \Psi(x_1, \dots, x_\ell)$$

with overwhelming probability over the coin tosses of all the algorithms. NCCrypt.Eval runs in time $\text{poly}(4^d, \ell, n, \log q)$.

Note that we stated the correctness of homomorphic evaluation on ciphertexts produced by the secret-key encryption algorithm NCCrypt.SecEnc . A similar lemma can be shown in the case of public-key encryption, if α is smaller by a factor of $\sqrt{n \log q}$. However, in our “optimal FHE” scheme in Section 4, we will only need to invoke this lemma with secret-key encryption.

Proof. It is easy to see that each step of the homomorphic evaluation algorithm, given by Eq. (5), simulates the execution of the branching program, given by Eq. (4). It remains to bound the noise growth during NCCrypt.Eval . We show this by induction.

In the sequel, we will abbreviate the noise function $\text{noise}_{s,\mu}(\mathbf{C})$ to $\text{noise}(\mathbf{C})$ since the secret key is fixed throughout the evaluation, and the message μ is clear from the context.

Clearly, $\text{noise}(\mathbf{V}_{0,i}) = 0$, since they $\mathbf{V}_{0,i}$ are just the messages, with no noise. Assume, as the inductive hypothesis, that for all $i \in \{1, 2, 3, 4, 5\}$,

$$\text{noise}(\mathbf{V}_{t-1,i}) = (t-1) \cdot \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q})$$

We will now bound $\text{noise}(\mathbf{V}_{t,i})$ for all $i \in \{1, 2, 3, 4, 5\}$. Note that

$$\begin{aligned} \text{noise}(\mathbf{V}_{t,i}) &\leq \text{noise}\left(\text{NCCrypt.ProtoMult}(\mathbf{V}_{t-1,\gamma_0}, \bar{\mathbf{C}}_{\text{var}(t)})\right) + \text{noise}\left(\text{NCCrypt.ProtoMult}(\mathbf{V}_{t-1,\gamma_1}, \mathbf{C}_{\text{var}(t)})\right) \\ &\leq |1 - x_{\text{var}(t)}| \cdot \text{noise}(\mathbf{V}_{t-1,\gamma_0}) + |x_{\text{var}(t)}| \cdot \text{noise}(\mathbf{V}_{t-1,\gamma_1}) + \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) \end{aligned}$$

where the second inequality holds by Claim 3.4.2 since all the ciphertexts encrypt bits, $\mathbf{C}_{\text{var}(t)}$ is a fresh secret-key encryption, and $\bar{\mathbf{C}}_{\text{var}(t)}$ contains exactly the same noise as $\mathbf{C}_{\text{var}(t)}$.

Since exactly one of $x_{\text{var}(t)}$ and $1 - x_{\text{var}(t)}$ is non-zero, we have

$$\begin{aligned} \text{noise}(\mathbf{V}_{t,i}) &\leq \max(\text{noise}(\mathbf{V}_{t-1,\gamma_0}), \text{noise}(\mathbf{V}_{t-1,\gamma_1})) + \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) \\ &\leq t \cdot \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) \end{aligned}$$

by the inductive hypothesis.

Thus, in particular,

$$\text{noise}(\mathbf{V}_\Psi) = \text{noise}(\mathbf{V}_{L,1}) \leq 4^d \cdot \tilde{O}_\kappa(\alpha q \cdot \sqrt{n \log q}) < q/8$$

by our setting of the parameter α . □

3.4 Achieving Fully Homomorphic Encryption

We know by Lemma 3.4 that the depth of the decryption circuit of NCCrypt is $c_{\text{dec}} \cdot \log(n \log q) = c_{\text{dec}} \log N$ for some constant $c_{\text{dec}} > 0$. Setting the depth $d = c \log N$ for some constant $c > c_{\text{dec}}$ in Lemma 3.6 and $\alpha \leq 1/\tilde{\Theta}_\kappa(4^d \cdot \sqrt{n \log q})$ gives us a bootstrappable encryption scheme. By the bootstrapping theorem (Theorem 2.14), this can be turned into a leveled FHE scheme, without additional assumptions. We state this theorem below:

Theorem 3.7. *Let n be the LWE dimension, q be the LWE modulus, $N := (n + 1) \cdot \lceil \log q \rceil$, and let $c > c_{\text{dec}}$ be a large enough constant (where c_{dec} is the decryption depth constant from Proposition 3.4), and*

$$\alpha \leq 1/\tilde{\Theta}_\kappa((n \log q)^{2c+1/2})$$

Then, there is a leveled FHE scheme that is secure under the decisional LWE assumption $\text{DLWE}_{n,q,\alpha}$.

In the next section, we will use a variant of the dimension-modulus reduction of [BV11], the effect of which will be to reduce the constant c above to a very small $\epsilon \rightarrow 0$, thus achieving a value of α that matches the best known lattice-based PKE schemes.

4 Successive Dimension-Modulus Reduction

In this section we revisit the dimension-modulus reduction technique from [BV11] and show that by successive application of this technique, we can achieve comparable lattice approximation factor to the best known factor for public key encryption.

We start by revisiting [BV11]’s dimension-modulus reduction in Section 4.1, and then proceed in Section 4.2 to present a bootstrappable homomorphic encryption scheme that is based on $\tilde{O}(n^{1+\epsilon} \cdot \sqrt{n \log(q)})$ -approximate GapSVP.

4.1 Dimension-Modulus Reduction (Revisited)

In the functions below, q is an integer and χ is a distribution over \mathbb{Z} :

- **SwitchKeyGen $_{q;p,\chi}(\mathbf{s}, \mathbf{t})$:** For a “source” key $\mathbf{s} \in \mathbb{Z}^{n_s}$ and “target” key $\mathbf{t} \in \mathbb{Z}^{n_t}$, we define a set of parameters that allow to switch ciphertexts under \mathbf{s} into ciphertexts under $(1, \mathbf{t})$.

Let $\hat{n}_s \triangleq n_s \cdot \lceil \log q \rceil$ be the dimension of $\text{PowersOfTwo}_q(\mathbf{s})$. Sample a uniform matrix $\mathbf{A}_{\mathbf{s}:\mathbf{t}} \xleftarrow{\$} \mathbb{Z}_p^{\hat{n}_s \times n_t}$ and a noise vector $\mathbf{e}_{\mathbf{s}:\mathbf{t}} \xleftarrow{\$} \chi^{\hat{n}_s}$. The function’s output is a matrix

$$\mathbf{P}_{\mathbf{s}:\mathbf{t}} = [\mathbf{b}_{\mathbf{s}:\mathbf{t}} \parallel -\mathbf{A}_{\mathbf{s}:\mathbf{t}}] \in \mathbb{Z}_p^{\hat{n}_s \times (n_t+1)},$$

where

$$\mathbf{b}_{\mathbf{s}:\mathbf{t}} := \left[\mathbf{A}_{\mathbf{s}:\mathbf{t}} \cdot \mathbf{t} + \mathbf{e}_{\mathbf{s}:\mathbf{t}} + \lfloor (p/q) \cdot \text{PowersOfTwo}_q(\mathbf{s}) \rfloor_G \right]_p \in \mathbb{Z}_p^{\hat{n}_s}.$$

Here, $\lfloor \cdot \rfloor_G$ is the Gaussian rounding procedure from Corollary 2.1.

- **SwitchKey $_q(\mathbf{P}_{\mathbf{s}:\mathbf{t}}, \mathbf{c}_s)$:** To switch a source ciphertext $\mathbf{c}_s \in \mathbb{Z}_q^{n_s}$ from a secret key \mathbf{s} to $(1, \mathbf{t})$, output

$$\mathbf{c}_t := [\mathbf{P}_{\mathbf{s}:\mathbf{t}}^T \cdot \text{BitDecomp}_q(\mathbf{c}_s)]_p \in \mathbb{Z}_p^{n_t+1}.$$

Lemma 4.1 (correctness). *Let $\mathbf{s} \in \mathbb{Z}^n$, $\mathbf{t} \in \mathbb{Z}^k$ be some vectors. Let χ be the discrete Gaussian $D_{\mathbb{Z}, \alpha p}$, and let $\mathbf{P}_{\mathbf{s}:\mathbf{t}} \leftarrow \text{SwitchKeyGen}_{q:p,\chi}(\mathbf{s}, \mathbf{t})$ and $\mathbf{P}_{\text{rand}} \leftarrow \text{RandParam}_{D_{\mathbb{Z}, \beta q}}(\mathbf{s})$. Let $\mathbf{c}_s \in \mathbb{Z}_q^n$ and let $\mathbf{c}'_s \leftarrow \text{Rand}(\mathbf{P}_{\text{rand}}, \mathbf{c}_s)$. Finally, set $\mathbf{c}_t \leftarrow \text{SwitchKey}(\mathbf{P}_{\mathbf{s}:\mathbf{t}}, \mathbf{c}'_s)$. Then there exists δ such that*

$$(p/q) \cdot \langle \mathbf{c}_s, \mathbf{s} \rangle - \delta = \langle \mathbf{c}_t, (1, \mathbf{t}) \rangle \pmod{p},$$

and $|\delta| < \tilde{O}_\kappa(\sqrt{n \log(q)}) \cdot \alpha p$ with all but $\text{negl}(\kappa)$ probability (over the coins in the experiment, and regardless of the generation of $\mathbf{s}, \mathbf{t}, \mathbf{c}_s$).

Proof. We expand the expression for $\langle \mathbf{c}_t, (1, \mathbf{t}) \rangle$:

$$\begin{aligned} \langle \mathbf{c}_t, (1, \mathbf{t}) \rangle &= \langle \mathbf{P}_{\mathbf{s}:\mathbf{t}}^T \cdot \text{BitDecomp}_q(\mathbf{c}'_s), (1, \mathbf{t}) \rangle \\ &= \langle \text{BitDecomp}_q(\mathbf{c}'_s), \mathbf{P}_{\mathbf{s}:\mathbf{t}} \cdot (1, \mathbf{t}) \rangle \\ &= \langle \text{BitDecomp}_q(\mathbf{c}'_s), \mathbf{e}_{\mathbf{s}:\mathbf{t}} + \lfloor (p/q) \cdot \text{PowersOfTwo}_q(\mathbf{s}) \rfloor_G \rangle. \end{aligned}$$

It follows that $\delta = \delta_1 + \delta_2$ where

$$\delta_1 = \langle \text{BitDecomp}_q(\mathbf{c}'_s), \mathbf{e}_{\mathbf{s}:\mathbf{t}} \rangle,$$

which, by Lemma 2.11, is bounded by $|\delta_1| \leq \tilde{O}_\kappa(\sqrt{n \log(q)}) \alpha p$ with all but $\text{negl}(\kappa)$ probability; and

$$\begin{aligned} \delta_2 &= \langle \text{BitDecomp}_q(\mathbf{c}'_s), \lfloor (p/q) \cdot \text{PowersOfTwo}_q(\mathbf{s}) \rfloor_G \rangle - (p/q) \cdot \langle \mathbf{c}'_s, \mathbf{s} \rangle \\ &= \langle \text{BitDecomp}_q(\mathbf{c}'_s), \lfloor (p/q) \cdot \text{PowersOfTwo}_q(\mathbf{s}) \rfloor_G - (p/q) \cdot \text{PowersOfTwo}_q(\mathbf{s}) \rangle. \end{aligned}$$

Applying Lemma 2.11, we get that $|\delta_2| \leq \tilde{O}_\kappa(\sqrt{n \log(q)})$ with all but $\text{negl}(\kappa)$ probability. \square

Security follows in a straightforward manner, the proof is omitted.

Lemma 4.2 (security). *Let $\mathbf{s} \in \mathbb{Z}^{n_s}$ be any vector. If we generate $\mathbf{t} \xleftarrow{\$} \mathbb{Z}_p^k$ and $\mathbf{P} \leftarrow \text{SwitchKeyGen}_{q:p,\chi}(\mathbf{s}, \mathbf{t})$, then \mathbf{P} is computationally indistinguishable from uniform over $\mathbb{Z}_p^{\hat{n}_s \times (n_t+1)}$, assuming the decisional LWE assumption $\text{DLWE}_{k,p,\chi}$.*

4.2 A Bootstrappable Scheme

Let $q : \mathbb{N} \rightarrow \mathbb{N}$ be a monotone function such that $q(n) \leq 2^n$ for all n , and $\alpha : \mathbb{N} \rightarrow \mathbb{R}$. Let $\chi(n)$ denote the discrete Gaussian distribution $D_{\mathbb{Z}, \alpha(n)q(n)}$. Finally, let $\epsilon > 0$.

The typical value of $\alpha(n)$ will be $1/\tilde{O}_\kappa(\sqrt{n \log(q)} \cdot n^\epsilon)$, where κ is the security parameter. As to the function $q(n)$, we will be interested in two ranges of parameters: In the first we will set $q(n)$ such that $\alpha(n) \cdot q(n) \approx \sqrt{n}$ (i.e. we set q such that $q(n) = \tilde{O}_\kappa(n^{1+\epsilon} \sqrt{\log(q)})$). This is the minimal q that allows to apply worst-case to average-case reductions for LWE . The second case is where $q(n) = 2^{n/2}$, which is the minimal q that allows to apply *classical* worst-case to average case reductions to the GapSVP problem.

The scheme DimReduced is defined as follows.

- **DimReduced.Keygen**(1^k): Define $n \triangleq (k \log(q(k)))^{2c_{\text{dec}}/\epsilon}$ (namely, $n^\epsilon = 4^{c_{\text{dec}} \cdot \log(k \log(q(k)))}$), where c_{dec} is as in Proposition 3.4. Assume for convenience that $n = k^{(1+\epsilon)^L}$ for some $L \in \mathbb{N}$ (otherwise round n to the next power up). Further define $k_i = k^{(1+\epsilon)^i}$, so $k_0 = k$ and $k_L = n$, and define $q_i = q(k_i)$. For convenience we denote $p \triangleq q_0$.

Let $(\text{ncksk}, \text{nckvk}, \text{nckpk}) \leftarrow \text{NCCrypt.Keygen}(1^n, q(n), 4^{c_{\text{dec}} \cdot \log(k \log(q(k))) + 1})$. Define $\mathbf{s}_L \triangleq \text{ncksk} \in \mathbb{Z}_{q_L}^{n_L}$.

For all $i = 0, \dots, L-1$, sample $\mathbf{s}_i \xleftarrow{\$} \mathbb{Z}_{q_i}^{n_i}$.

Next, we generate dimension-modulus switching parameters (see Section 4.1), and randomization parameters (see Section 2.4) for all $i \in [L]$:

$$\mathbf{P}_{i:(i-1)} \leftarrow \text{SwitchKeyGen}_{q_i:q_{i-1}}((1, \mathbf{s}_i), \mathbf{s}_{i-1}) ,$$

and

$$\mathbf{P}_{\text{rand},i} \leftarrow \text{RandParam}_{D_{\mathbb{Z}, \alpha(k_i)q_i}}(\mathbf{s}_i) .$$

Finally, output the keys $sk \triangleq (\mathbf{s}_0, \mathbf{s}_L)$, $pk \triangleq \text{nckpk}$, $evk \triangleq (\text{nckvk}, \{\mathbf{P}_{i:(i-1)}\}_{i \in [L]}, \{\mathbf{P}_{\text{rand},i}\}_{i \in [L]})$.

We note that as ϵ approaches 0, $n = k^{\Theta(1/\epsilon)}$ becomes larger. If k is proportional to the security parameter, then ϵ must be bounded by a constant to keep n polynomially bounded. We further note that $L = \Theta(\log(1/\epsilon)/\epsilon)$.

- **DimReduced.PubEnc** $_{pk}(\mu)$ / **DimReduced.SecEnc** $_{sk}(\mu)$: The asymmetric and symmetric encryption procedures are identical to **NCCrypt**. Since **DimReduced**'s public key and secret key contain those of **NCCrypt**, this can be done in a straightforward manner.
- **DimReduced.Eval** $_{evk}(f, \mathbf{C}_1, \dots, \mathbf{C}_t)$: To perform homomorphic evaluation, we first compute

$$\mathbf{C}_f \leftarrow \text{NCCrypt.Eval}_{\text{nckvk}}(f, \mathbf{C}_1, \dots, \mathbf{C}_t) .$$

We then consider $\mathbf{c}_f \in \{0, 1\}^{n \lceil \log(q_L) \rceil}$, which is the second row of \mathbf{C}_f . We set $\mathbf{c}_L := \text{Combine}_{q_L}(\mathbf{c}_f)$.

We then compute, in order for $i = L-1, \dots, 0$, the ciphertexts $\mathbf{c}_{\text{rand},i+1} \leftarrow \text{Rand}(\mathbf{P}_{\text{rand},i+1}, \mathbf{c}_{i+1})$, and then $\mathbf{c}_i \leftarrow \text{SwitchKey}(\mathbf{P}_{(i+1):i}, \mathbf{c}_{\text{rand},i+1})$. Finally, $\mathbf{c}_0 \in \mathbb{Z}_p^k$ is output as the final ciphertext.

- **DimReduced.Dec** $_{sk}(\mathbf{c})$: We recall that $\mathbf{c} \in \mathbb{Z}_p^k$. We output $\mu^* = 0$ if $\left| [\langle \mathbf{c}, (1, \mathbf{s}_0) \rangle]_p \right| < p/8$, and $\mu^* = 1$ otherwise.

Security is stated in the next lemma and follows immediately using a hybrid argument and using the security properties of the scheme **NCCrypt**, the ciphertext randomization procedure (Lemma 2.9) and the dimension-modulus reduction procedure (Lemma 4.2). The formal proof is omitted.

Lemma 4.3. *The scheme **DimReduced** is secure under the $\text{DLWE}_{k,q(k),\alpha(k)}$ assumption.*

Correctness poses a more challenging task. We want to prove that **DimReduced** can homomorphically evaluate an augmented decryption circuit. Proving this when $\alpha(n)$ is small (e.g. $\alpha(n) = 1/n^3$) is fairly easy. However, since we wish to achieve optimal parameters, the analysis is more involved and appears in the following lemma.

We define $\tau \triangleq 2^{\lceil \log(q_L) \rceil - 2} / q_L$ and notice that $\tau \in (1/4, 1/2]$.

Lemma 4.4. Let $\alpha(n) = 1/\tilde{O}_\kappa(\sqrt{n \log(q)} \cdot n^\epsilon)$, and let $q(n) \geq \tilde{O}(\sqrt{n}/\alpha(n))$. Consider a set of keys generated by $(sk, pk, evk) \leftarrow \text{DimReduced.Keygen}(1^k)$, and recall that $sk = (s_0, s_L)$. Let $\mathbf{K}_i \leftarrow \text{DimReduced.SecEnc}_{sk}(\text{BitDecomp}(s_0)[i])$. Namely, \mathbf{K}_i is the symmetric encryption of the i th bit of s_0 .

Let f be an augmented decryption circuit as per Definition 2.13. Namely, let $\mathbf{c}, \mathbf{c}' \in \mathbb{Z}_p^k$ and $\mu, \mu' \in \{0, 1\}$ be such that

$$[\langle \mathbf{c}, \mathbf{s}_0 \rangle]_p = \mu \cdot \tau p + e ,$$

where $|e| < p/8$, and similarly for \mathbf{c}' . The function f is the function that on input x , treats x as a secret key and decrypts \mathbf{c}, \mathbf{c}' , and outputs the NAND of their decryptions.

Let

$$\mathbf{c}_0 \leftarrow \text{DimReduced.Eval}_{evk}(f, \mathbf{K}_1, \mathbf{K}_2, \dots) ,$$

and note that this is syntactically well defined since f can be represented as a Boolean circuit of depth $c_{\text{dec}} \cdot \log(k \log(q(k))) + 1$.

Then with all but $\text{negl}(\kappa)$ probability,

$$[\langle \mathbf{c}_0, \mathbf{s}_0 \rangle]_p = \mu^* \cdot \tau p + e_f ,$$

where $\mu^* = \overline{(\mu \wedge \mu')}$, $|e_f| < p/8$ are as above.

Proof. Consider the process of execution of $\text{DimReduced.Eval}_{evk}(f, \mathbf{K}_1, \mathbf{K}_2, \dots)$. It starts by generating \mathbf{c}_L , where we are guaranteed by the correctness of NCCrypt that

$$[\langle \mathbf{c}_L, \mathbf{s}_L \rangle]_{q_L} = \mu^* \cdot \tau q_L + e_L ,$$

where

$$|e_L|/q_L \leq \tilde{O}_\kappa(\sqrt{n \log(q_L)} \cdot 4^{c_{\text{dec}} \cdot \log(k \log(q(k))) + 1}) \alpha(n) = \tilde{O}_\kappa(n^\epsilon \cdot \sqrt{n \log(q_L)}) \cdot \alpha(n) ,$$

and we will set $\alpha(n) = 1/(\sqrt{n \log(q)} \cdot n^\epsilon \text{polylog}(\kappa))$ with sufficiently large polylogarithmic factor to offset the one coming from the noise so that

$$|e_L|/q_L \leq 1/\text{polylog}(\kappa) .$$

We then commence with $L = O(1)$ levels of randomization followed by modulus-dimension reduction. Let us consider the effect of these operations at level i .

Lemma 2.10 guarantees that in the randomization step, the relative noise grows by an additive factor of at most $\tilde{O}_\kappa(\sqrt{k_i \log(q_i)}) \cdot \alpha(k_i) = k_i^{-\epsilon} \cdot \tilde{O}_\kappa(1)/\text{polylog}(\kappa)$. Again, the idea is to define α with sufficiently large polylogarithmic factor to offset those coming from $\tilde{O}_\kappa(\cdot)$.

Lemma 4.1 guarantees that in the key switching step, the relative noise grows by an additive factor of $\tilde{O}_\kappa(\sqrt{k_i \log(q_i)}) \cdot \alpha(k_{i-1})$. We recall that $k_i = k_{i-1}^{1+\epsilon}$ and that $q(n) < 2^n$. Therefore

$$\sqrt{k_i \log(q_i)} \leq \sqrt{(k_{i-1} \log(q_{i-1}))^{1+\epsilon}} \leq \sqrt{(k_{i-1} \log(q_{i-1}))} \cdot k_{i-1}^\epsilon .$$

Therefore, setting the polylogarithmic factors right, we get an additive relative error of at most $1/\text{polylog}(\kappa)$.

Putting all of these together, we get that

$$|e_f| \leq L/\text{polylog}(\kappa) \ll p/8 ,$$

and the result follows. \square

Finally, we derive the worst-case lattice approximation factor based on Corollary 2.6. We recall that a bootstrappable homomorphic encryption scheme implies a leveled FHE scheme under the same assumptions, and a pure FHE scheme with an additional circular security assumption.

Corollary 4.5. *For all $\epsilon > 0$, there exist:*

- *A bootstrappable homomorphic encryption scheme based on the worst-case quantum hardness of solving $\text{GapSVP}_{\tilde{O}(n^{1.5+\epsilon})}$ and $\text{SIVP}_{\tilde{O}(n^{1.5+\epsilon})}$.*
- *A bootstrappable homomorphic encryption scheme based on the worst-case classical hardness of solving $\text{GapSVP}_{\tilde{O}(n^{2+\epsilon})}$.*

The first (quantum) case is derived from Lemma 4.4 by setting $q(n) = \sqrt{n}/\alpha(n) = \text{poly}(n)$, and the second (classical) case is derived by setting $q(n) = 2^{n/2}$.

Improving Key and Ciphertext sizes. The scheme `DimReduced` uses a ladder of `LWE` instances, ranging from short (k, p) to polynomially larger (n, q) . In the description above, the public key of the scheme is derived from that of `NCCrypt`, and therefore depends on n and not on k . Likewise, the “input ciphertexts” (the ones before homomorphic evaluation) also depend on n .

We note here that this can be fixed in such a way that only the *evk* depends on n , and the rest of the parameters are exactly the same as Regev’s scheme with parameters (k, p) . This is done in a standard way (used e.g. in [BV11]) as follows.

We will generate the public key as a standard Regev public key with parameters (k, p) , and in the evaluation key we will encrypt the bits of the respective secret key using `NCCrypt`. This will allow to perform homomorphic operations by evaluating the augmented decryption circuit. Namely, the ciphertexts visible to the user of the scheme will always be short, but in the process of homomorphic evaluation, larger ciphertexts are used to accommodate the homomorphic operation, and once it is done dimension-modulus reduction will be used to shrink the output ciphertext back to the original size. Since this is standard practice, we omit the technical description.

References

- [AD97] Miklós Ajtai and Cynthia Dwork. A public-key cryptosystem with worst-case/average-case equivalence. In Frank Thomson Leighton and Peter W. Shor, editors, *STOC*, pages 284–293. ACM, 1997.
- [Bar89] David A. Mix Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc^1 . *J. Comput. Syst. Sci.*, 38(1):150–164, 1989.
- [BDFP86] Allan Borodin, Danny Dolev, Faith E. Fich, and Wolfgang J. Paul. Bounds for width two branching programs. *SIAM J. Comput.*, 15(2):549–560, 1986.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS*, pages 309–325. ACM, 2012. Invited to ACM Transactions on Computation Theory.
- [BLP⁺13] Zvika Brakerski, Adeline Langlois, Chris Peikert, Oded Regev, and Damien Stehlé. Classical hardness of learning with errors. *CoRR*, abs/1306.0281, 2013. Preliminary version in STOC 2013.

- [Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *FOCS*, pages 97–106. IEEE, 2011. Invited to SIAM Journal on Computing.
- [Gen09a] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. <http://crypto.stanford.edu/craig>.
- [Gen09b] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [Gen10] Craig Gentry. Toward basing fully homomorphic encryption on worst-case hardness. In *CRYPTO*, pages 116–137, 2010.
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Cynthia Dwork, editor, *STOC*, pages 197–206. ACM, 2008.
- [GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. *IACR Cryptology ePrint Archive*, 2013:340, 2013. Preliminary version in CRYPTO 2013.
- [IP07] Yuval Ishai and Anat Paskin. Evaluating branching programs on encrypted data. In Salil P. Vadhan, editor, *TCC*, volume 4392 of *Lecture Notes in Computer Science*, pages 575–594. Springer, 2007.
- [MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of lwe search-to-decision reductions. In Phillip Rogaway, editor, *CRYPTO*, volume 6841 of *Lecture Notes in Computer Science*, pages 465–484. Springer, 2011.
- [MP11] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. *IACR Cryptology ePrint Archive*, 2011:501, 2011. Extended abstract in Eurocrypt 2012.
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In Michael Mitzenmacher, editor, *STOC*, pages 333–342. ACM, 2009.
- [Reg04] Oded Regev. New lattice-based cryptographic constructions. *J. ACM*, 51(6):899–942, 2004.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 84–93. ACM, 2005. Full version in [Reg09].
- [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.
- [Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.

Machine Learning Classification over Encrypted Data

Raphaël Bost* Raluca Ada Popa† Stephen Tu‡ Shafi Goldwasser‡

Abstract

Machine learning classification is used in numerous settings nowadays, such as medical or genomics predictions, spam detection, face recognition, and financial predictions. Due to privacy concerns, in some of these applications, it is important that the data and the classifier remain confidential.

In this work, we construct three major classification protocols that satisfy this privacy constraint: hyperplane decision, Naïve Bayes, and decision trees. We also enable these protocols to be combined with AdaBoost. At the basis of these constructions is a new library of building blocks for constructing classifiers securely; we demonstrate that this library can be used to construct other classifiers as well, such as a multiplexer and a face detection classifier.

We implemented and evaluated our library and classifiers. Our protocols are efficient, taking milliseconds to a few seconds to perform a classification when running on real medical datasets.

1 Introduction

Classifiers are an invaluable tool for many tasks today, such as medical or genomics predictions, spam detection, face recognition, and finance. Many of these applications handle sensitive data [WGH12, SG11, SG13], so it is important that the data and the classifier remain private.

Consider the typical setup of supervised learning, depicted in Figure 1. Supervised learning algorithms consist of two phases: (i) the training phase during which the algorithm learns a model w from a data set of labeled examples, and (ii) the classification phase that runs a classifier C over a previously unseen feature vector x , using the model w to output a prediction $C(x, w)$.

In applications that handle sensitive data, it is important that the feature vector x and the model w remain secret to one or some of the parties involved. Consider the example of a medical study or a hospital having a model built out of the private medical profiles of some patients; the model is sensitive because it can leak information about the patients, and its usage has to be HIPAA¹ compliant. A client wants to use the model to make a prediction about her health (e.g., if she is likely to contract a certain disease, or if she would be treated successfully at the hospital), but does not want to reveal her sensitive medical profile. Ideally, the hospital and the client run a protocol at the end of which the client learns one bit (“yes/no”), and neither party learns anything else about the other party’s input. A similar setting arises for a financial institution (e.g., an insurance company) holding a sensitive model, and a customer wanting to estimate rates or quality of service based on her personal information.

Throughout this paper, we refer to this goal shortly as *privacy-preserving classification*. Concretely, a client has a private input represented as a feature vector x , and the server has a private input consisting of a private model w . The way the model w is obtained is independent of our protocols here. For example, the server could have computed the model w after running the training phase on plaintext data as usual. Only the classification needs to be privacy-preserving: the client should learn $C(x, w)$ but nothing else about the model w , while the server should not learn anything about the client’s input or the classification result.

*Direction Générale de l’Armement - Maitrise de l’Information. Work done while visiting MIT CSAIL. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DGA or the French Government.

†ETH Zürich

‡MIT CSAIL

¹Health Insurance Portability and Accountability Act of 1996

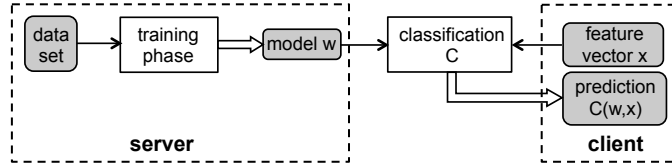


Figure 1: Model overview. Each shaded box indicates private data that should be accessible to only one party: the dataset and the model to the server, and the input and prediction result to the client. Each straight non-dashed rectangle indicates an algorithm, single arrows indicate inputs to these algorithms, and double arrows indicate outputs.

Machine learning algorithm	Classifier
Perceptron	Hyperplane decision
Least squares	Hyperplane decision
Fischer linear discriminant	Hyperplane decision
Support vector machine	Hyperplane decision
Naïve Bayes	Naïve Bayes
Decision trees (ID3/C4.5)	Decision trees

Table 1: Machine learning algorithms and their classifiers, defined in Section 3.1.

In this work, we construct efficient privacy-preserving protocols for three of the most common classifiers: hyperplane decision, Naïve Bayes, and decision trees, as well as a more general classifier combining these using AdaBoost. These classifiers are widely used – even though there are many machine learning algorithms, most of them end up using one of these three classifiers, as described in Table 1.

While generic secure multi-party computation [Yao82, GMW87, HKS⁺10, MNPS04, BDNP08] can implement any classifier in principle, due to their generality, such schemes are not efficient for common classifiers. As described in Section 10.5, on a small classification instance, such tools ([HKS⁺10, BDNP08]) ran out of memory on a powerful machine with 256GB of RAM; also, on an artificially simplified classification instance, these protocols ran ≈ 500 times slower than our protocols ran on the non-simplified instance.

Hence, protocols specialized to the classification problem promise better performance. However, most existing work in machine learning and privacy [LP00, DHC04, WY04, ZW05, BDMN05, VKC08, GLN12] focuses on preserving privacy during the *training phase*, and does not address classification. The few works on privacy-preserving classification either consider a weaker security setting in which the client learns the model [BLN13] or focus on specific classifiers (e.g., face detectors [EFG⁺09, SSW09, AB06, AB07]) that are useful in limited situations.

Designing efficient privacy-preserving classification faces two main challenges. The first is that the computation performed over sensitive data by some classifiers is quite complex (e.g., decision trees), making it hard to support efficiently. The second is providing a solution that is more generic than the three classifiers: constructing a separate solution for each classifier does not provide insight into how to combine these classifiers or how to construct other classifiers. Even though we contribute privacy-preserving protocols for three of the most common classifiers, various settings use other classifiers or use a combination of these three classifiers (e.g., AdaBoost). We address these challenges using two key techniques.

Our main technique is to identify *a set of core operations* over encrypted data that underlie many classification protocols. We found these operations to be comparison, argmax, and dot product. We use *efficient* protocols for each one of these, either by improving existing schemes (e.g., for comparison) or by constructing new schemes (e.g., for argmax).

Our second technique is to design these building blocks in a *composable* way, with regard to *both* functionality and security. To achieve this goal, we use a set of sub-techniques:

- The input and output of all our building blocks are data encrypted with additively homomorphic encryption. In addition, we provide a mechanism to switch from one encryption scheme to another. Intuitively, this enables a building block’s output to become the input of another building block;
- The API of these building blocks is flexible: even though each building block computes a fixed function, it allows

a choice of which party provides the inputs to the protocol, which party obtains the output of the computation, and whether the output is encrypted or decrypted;

- The security of these protocols composes using modular sequential composition [Can98].

We emphasize that the contribution of our building blocks library goes beyond the classifiers we build in this paper: a user of the library can construct other privacy-preserving classifiers in a modular fashion. To demonstrate this point, we use our building blocks to construct a multiplexer and a classifier for face detection, as well as to combine our classifiers using AdaBoost.

We then use these building blocks to construct novel privacy-preserving protocols for three common classifiers. Some of these classifiers incorporate additional techniques, such as an efficient evaluation of a decision tree with fully homomorphic encryption (FHE) based on a polynomial representation requiring only a small number of multiplications and based on SIMD FHE slots (see Section 7.2). All of our protocols are secure against passive adversaries (see Section 3.2.3).

We also provide an implementation and an evaluation of our building blocks and classifiers. We evaluate our classifiers on real datasets with private data about breast cancer, credit card approval, audiology, and nursery data; our algorithms are efficient, running in milliseconds up to a few seconds, and consume a modest amount of bandwidth.

The rest of the paper is organized as follows. Section 2 describes related work, Section 3 provide the necessary machine learning and cryptographic background, Section 4 presents our building blocks, Sections 5–8 describe our classifiers, and Sections 9–10 present our implementation and evaluation results.

2 Related work

Our work is the first to provide efficient privacy-preserving protocols for a broad class of classifiers.

Secure two-party computation protocols for generic functions exist in theory [Yao82, GMW87, LP07, IPS08, LP09] and in practice [HKS⁺10, MNPS04, BDNP08]. However, these rely on heavy cryptographic machinery, and applying them directly to our problem setting would be too inefficient as exemplified in Section 10.5.

Previous work focusing on privacy-preserving machine learning can be broadly divided into two categories: (i) techniques for privacy-preserving training, and (ii) techniques for privacy-preserving classification (recall the distinction from Figure 1). Most existing work falls in the first category, which we discuss in Section 2.1. Our work falls in the second category, where little work has been done, as we discuss in Section 2.2. We also mention work related to the building blocks we use in our protocols in Section 2.3.

It is worth mentioning that our work on privacy-preserving classification is complementary to work on differential privacy in the machine learning community (see *e.g.* [CMS11]). Our work aims to hide each user’s input data to the classification phase, whereas differential privacy seeks to construct classifiers/models from sensitive user training data that leak a bounded amount of information about each individual in the training data set.

2.1 Privacy-preserving training

A set of techniques have been developed for privacy-preserving *training* algorithms such as Naïve Bayes [VKC08, WY04, ZW05], decision trees [BDMN05, LP00], linear discriminant classifiers [DHC04], and more general kernel methods [LLM06].

Grapel *et al.* [GLN12] show how to train several machine learning classifiers using a somewhat homomorphic encryption scheme. They focus on a few simple classifiers (*e.g.* the linear means classifier), and do not elaborate on more complex algorithms such as support vector machines. They also support private classification, but in a weaker security model where the client learns more about the model than just the final sign of the classification. Indeed, performing the final comparison with fully homomorphic encryption (FHE) alone is inefficient, a difficulty we overcome with an interactive setting.

2.2 Privacy-preserving classification

Little work has been done to address the general problem of privacy-preserving classification in practice; previous work focuses on a weaker security setting (in which the client learns the model) and/or only supports specific classifiers.

In Bos *et al.* [BLN13], a third party can compute medical prediction functions over the encrypted data of a patient using fully homomorphic encryption. In their setting, everyone (including the patient) knows the predictive model, and their algorithm hides only the input of the patient from the cloud. Our protocols, on the other hand, also hide the model from the patient. Their algorithms cannot be applied to our setting because they leak more information than just the bit of the prediction to the patient. Furthermore, our techniques are notably different; using FHE directly for our classifiers would result in significant overheads.

Barni *et al.* [BFK⁺09, BFL⁺09] construct secure evaluation of linear branching programs, which they use to implement a secure classifier of ECG signals. Their technique is based on finely-tuned garbled circuits. By comparison, our construction is not limited to branching programs (or decision trees), and our evaluation shows that our construction is twice as fast on branching programs. In a subsequent work [BFL⁺11], Barni *et al.* study secure classifiers based on neural networks, which is a generalization of the perceptron classifiers, and hence also covered by our work.

Other works [EFG⁺09, SSW09, AB06, AB07] construct specific face recognition or detection classifiers. We focus on providing a set of *generic* classifiers and building blocks to construct more complex classifiers. In Section 10.1.2, we show how to construct a private face detection classifier using the modularity of our techniques.

2.3 Work related to our building blocks

Two of the basic components we use are private comparison and private computation of dot products. These items have been well-studied previously; see [Yao82, DGK07, DGK09, Veu11, LT05, AB06, KSS09] for comparison techniques and [AD01, GLLM04, Kil05, AB06] for techniques to compute dot products. Section 4.1 discusses how we build on these tools.

3 Background and preliminaries

3.1 Classification in machine learning algorithms

The user's input x is a vector of d elements $x = (x_1, \dots, x_d) \in \mathbb{R}^d$, called a feature vector. To classify the input x means to evaluate a classification function $C_w : \mathbb{R}^d \mapsto \{c_1, \dots, c_k\}$ on x . The output is $c_{k^*} = C_w(x)$, where $k^* \in \{1 \dots k\}$; c_{k^*} is the class to which x corresponds, based on the model w . For ease of notation, we often write k^* instead of c_{k^*} , namely $k^* = C_w(x)$.

We now describe how three popular classifiers work on regular, unencrypted data. These classifiers differ in the model w and the function C_w . For more details, we refer the reader to [BN06].

Hyperplane decision-based classifiers. For this classifier, the model w consists of k vectors in \mathbb{R}^d ($w = \{w_i\}_{i=1}^k$). The classifier is (cf. [BN06]):

$$k^* = \operatorname{argmax}_{i \in [k]} \langle w_i, x \rangle, \quad (1)$$

where $\langle w_i, x \rangle$ denotes inner product between w_i and x .

We now explain how Eq. (1) captures many common machine learning algorithms. A hyperplane based classifier typically works with a hypothesis space \mathcal{H} equipped with an inner product $\langle \cdot, \cdot \rangle$. This classifier usually solves a binary classification problem ($k = 2$): given a user input x , x is classified in class c_2 if $\langle w, \phi(x) \rangle \geq 0$, otherwise it is labeled as part of class c_1 . Here, $\phi : \mathbb{R}^d \mapsto \mathcal{H}$ denotes the feature mapping from \mathbb{R}^d to \mathcal{H} [BN06]. In this work, we focus on the case when $\mathcal{H} = \mathbb{R}^d$ and note that a large class of infinite dimensional spaces can be approximated with a finite dimensional space (as in [RR07]), including the popular gaussian kernel (RBF). In this case, $\phi(x) = x$ or $\phi(x) = Px$ for a randomized projection matrix P chosen during training. Notice that Px consists solely of inner products; we will show how to support private evaluation of inner products later, so for simplicity we drop P from the discussion. To extend such a classifier from 2 classes to k classes, we use one of the most common approaches, *one-versus-all*, where k different models $\{w_i\}_{i=1}^k$ are trained to discriminate each class from all the others. The decision rule is then given by

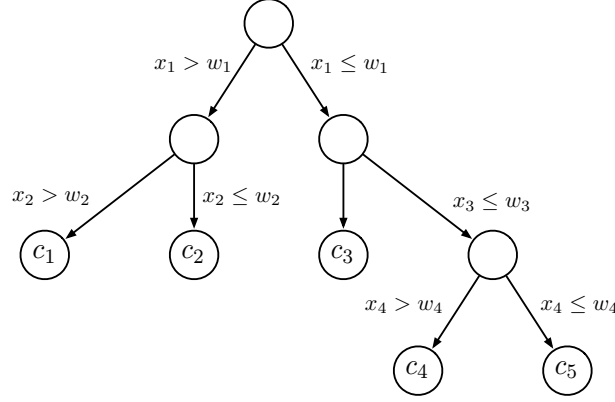


Figure 2: Decision tree

(cf. [BN06]) to be Eq. (1). This framework is general enough to cover many common algorithms, such as support vector machines (SVMs), logistic regression, and least squares.

Naïve Bayes classifiers. For this classifier, the model w consists of various probabilities: the probability that each class c_i occurs, namely $\{p(C = c_i)\}_{i=1}^k$, and the probabilities that an element x_j of x occurs in a certain class c_i . More concretely, the latter is the probability of the j -th component x_j of x to be v when x belongs to category c_i ; this is denoted by $\{\{p(X_j = v|C = c_i)\}_{v \in D_j}\}_{j=1}^d\}_{i=1}^k$, where D_j is X_j 's domain². The classification function, using a *maximum a posteriori* decision rule, works by choosing the class with the highest posterior probability:

$$\begin{aligned}
 k^* &= \operatorname{argmax}_{i \in [k]} p(C = c_i | X = x) \\
 &= \operatorname{argmax}_{i \in [k]} p(C = c_i, X = x) \\
 &= \operatorname{argmax}_{i \in [k]} p(C = c_i, X_1 = x_1, \dots, X_d = x_d)
 \end{aligned}$$

where the second equality follows from applying Bayes' rule (we omitted the normalizing factor $p(X = x)$ because it is the same for a fixed x).

The Naïve Bayes model assumes that $p(C = c_i, X = x)$ has the following factorization:

$$\begin{aligned}
 &p(C = c_i, X_1 = x_1, \dots, X_d = x_d) \\
 &= p(C = c_i) \prod_{j=1}^d p(X_j = x_j | C = c_i),
 \end{aligned}$$

namely, each of the d features are conditionally independent given the class. For simplicity, we assume that the domain of the features values (the x_i 's) is discrete and finite, so the $p(X_j = x_j | C = c_i)$'s are probability masses.

Decision trees. A decision tree is a non-parametric classifier which works by partitioning the feature vector space one attribute at a time; interior nodes in the tree correspond to partitioning rules, and leaf nodes correspond to class labels. A feature vector x is classified by walking the tree starting from the root, using the partitioning rule at each node to decide which branch to take until a leaf node is encountered. The class at the leaf node is the result of the classification.

Figure 2 gives an example of a decision tree. The model consists of the structure of the tree and the decision criteria at each node (in this case the thresholds w_1, \dots, w_4).

²Be careful to distinguish between X_j , the probabilistic random variable representing the values taken by the j -th feature of user's input, and x_j , the actual value taken by the specific vector x .

3.2 Cryptographic preliminaries

3.2.1 Cryptosystems

In this work, we use three additively homomorphic cryptosystems. A public-key encryption scheme HE is additively homomorphic if, given two encrypted messages $\text{HE.Enc}(a)$ and $\text{HE.Enc}(b)$, there exists a public-key operation \oplus such that $\text{HE.Enc}(a) \oplus \text{HE.Enc}(b)$ is an encryption of $a + b$. We emphasize that these are homomorphic *only for addition*, which makes them efficient, unlike fully homomorphic encryption [Gen09], which supports any function. The cryptosystems we use are:

1. the QR (Quadratic Residuosity) cryptosystem of Goldwasser-Micali [GM82],
2. the Paillier cryptosystem [Pai99], and
3. a leveled fully homomorphic encryption (FHE) scheme, HELib [Hal13]

3.2.2 Cryptographic assumptions

We prove that our protocols are secure based on the semantic security [Gol04] of the above cryptosystems. These cryptosystems rely on standard and well-studied computational assumptions: the Quadratic Residuosity assumption, the Decisional Composite Residuosity assumption, and the Ring Learning With Error (RLWE) assumption.

3.2.3 Adversarial model

We prove security of our protocols using the secure two-party computation framework for passive adversaries (or honest-but-curious [Gol04]) defined in Appendix B.1. To explain what a passive adversary is, at a high level, consider that a party called party A is compromised by such an adversary. This adversary tries to learn as much private information about the input of the other party by watching all the information party A receives; nevertheless, this adversary cannot prevent party A from following the prescribed protocol faithfully (hence, it is not an active adversary).

To enable us to compose various protocols into a bigger protocol securely, we invoke modular sequential composition (see Appendix B.2).

3.3 Notation

All our protocols are between two parties: parties A and B for our building blocks and parties C (client) and S (server) for our classifiers.

Inputs and outputs of our building blocks are either unencrypted or encrypted with an additively homomorphic encryption scheme. We use the following notation. The plaintext space of QR is \mathbb{F}_2 (bits), and we denote by $[b]$ a bit b encrypted under QR; the plaintext space of Paillier is \mathbb{Z}_N where N is the public modulus of Paillier, and we denote by $\llbracket m \rrbracket$ an integer m encrypted under Paillier. The plaintext space of the FHE scheme is \mathbb{F}_2 . We denote by SK_P and PK_P , a secret and a public key for Paillier, respectively. Also, we denote by SK_{QR} and PK_{QR} , a secret and a public key for QR.

For a constant b , $a \leftarrow b$ means that a is assigned the value of b . For a distribution \mathcal{D} , $a \leftarrow \mathcal{D}$ means that a gets a sample from \mathcal{D} .

4 Building blocks

In this section, we develop a library of building blocks, which we later use to build our classifiers. We designed this library to also enable constructing other classifiers than the ones described in our paper. The building blocks in this section combine existing techniques with either new techniques or new optimizations.

Type	Input A	Input B	Output A	Output B	Implementation
1	PK_P, PK_{QR}, a	SK_P, SK_{QR}, b	$[a < b]$	–	Sec. 4.1.1
2	$PK_P, SK_{QR}, \llbracket a \rrbracket, \llbracket b \rrbracket$	SK_P, PK_{QR}	–	$[a \leq b]$	Sec. 4.1.2
3	$PK_P, SK_{QR}, \llbracket a \rrbracket, \llbracket b \rrbracket$	SK_P, PK_{QR}	$a \leq b$	$[a \leq b]$	Sec. 4.1.2
4	$PK_P, PK_{QR}, \llbracket a \rrbracket, \llbracket b \rrbracket$	SK_P, SK_{QR}	$[a \leq b]$	–	Sec. 4.1.3
5	$PK_P, PK_{QR}, \llbracket a \rrbracket, \llbracket b \rrbracket$	SK_P, SK_{QR}	$[a \leq b]$	$a \leq b$	Sec. 4.1.3

Table 2: The API of our comparison protocol and its implementation. There are five types of comparisons each having a different setup.

4.1 Comparison

We now describe our comparison protocol. In order for this protocol to be used in a wide range of classifiers, its setup needs to be *flexible*: namely, it has to support a range of choices regarding which party gets the input, which party gets the output, and whether the input or output are encrypted or not. Table 2 shows the various ways our comparison protocol can be used. In each case, each party learns *nothing else* about the other party’s input other than what Table 2 indicates as the output.

We implemented each row of Table 2 by modifying existing protocols. We explain only the modifications here, and defer full protocol descriptions to Appendix A and proofs of security to Appendix C.1.

There are at least two approaches to performing comparison efficiently: using specialized homomorphic encryption [DGK07, DGK09, EFG⁺09, Veu11], or using garbled circuits [BHKR13]. We compared empirically the performance of these approaches and concluded that the former is more efficient for comparison of encrypted values, and the second is more efficient for comparison of unencrypted values.

4.1.1 Comparison with unencrypted inputs (Row 1)

To compare unencrypted inputs, we use garbled circuits implemented with the state-of-the-art garbling scheme of Bellare et al. [BHKR13], the short circuit for comparison of Kolesnikov et al. [KSS09] and a well-known oblivious transfer (OT) scheme due to Naor and Pinkas [NP01]. Since most of our other building blocks expect inputs encrypted with homomorphic encryption, one also needs to convert from a garbled output to homomorphic encryption to enable composition. We can implement this easily using the random shares technique in [KSS13].

The above techniques combined give us the desired comparison protocol. Actually, we can directly combine them to build an even more efficient protocol: we use an enhanced comparison circuit that also takes as input a masking bit. Using a garbled circuit and oblivious transfer, A will compute $(a < b) \oplus c$ where c is a bit randomly chosen by B. B will also provide an encryption $[c]$ of c , enabling A to compute $[a < b]$ using the homomorphic properties of QR.

4.1.2 Comparison with encrypted inputs (Rows 2, 3)

Our classifiers also require the ability to compare two encrypted inputs. More specifically, suppose that party A wants to compare two encrypted integers a and b , but party B holds the decryption key. To implement this task, we slightly modify Veugen’s [Veu11] protocol: it uses a comparison with unencrypted inputs protocol as a sub-procedure, and we replaced it with the comparison protocol we just described above. This yields a protocol for the setup in Row 2. To ensure that A receives the plaintext output as in Row 3, B sends the encrypted result to A who decrypts it. Appendix A provides the detailed protocol.

4.1.3 Reversed comparison over encrypted data (Row 4, 5)

In some cases, we want the result of the comparison to be held by the party that does not hold the encrypted data. For this, we modify Veugen’s protocol to reverse the outputs of party A and party B: we achieve this by exchanging the role of party A and party B in the last few steps of the protocol, after invoking the comparison protocol with unencrypted inputs. We do not present the details in the paper body because they are not insightful, and instead include them in Appendix A.

This results in a protocol whose specification is in Row 4. To obtain Row 5, A sends the encrypted result to B who can decrypt it.

4.1.4 Negative integers comparison and sign determination

Negative numbers are handled by the protocols above unchanged. Even though the Paillier plaintext size is “positive”, a negative number simply becomes a large number in the plaintext space due to cyclicity of the space. As long as the values encrypted are within a preset interval $(-2^\ell, 2^\ell)$ for some fixed ℓ , Veugen’s protocol and the above protocols work correctly.

In some cases, we need to compute the sign of an encrypted integer $\llbracket b \rrbracket$. In this case, we simply compare to the encryption of 0.

4.2 argmax over encrypted data

In this scenario, party A has k values a_1, \dots, a_k encrypted under party B ’s secret key and wants party B to know the argmax over these values (the index of the largest value), but neither party should learn anything else. For example, if A has values $\llbracket 1 \rrbracket$, $\llbracket 100 \rrbracket$ and $\llbracket 2 \rrbracket$, B should learn that the second is the largest value, but learn nothing else. In particular, B should not learn the order relations between the a_i ’s.

Our protocol for argmax is shown in Protocol 1. We now provide intuition into the protocol and its security.

Intuition. Let’s start with a strawman. To prevent B from learning the order of the k values $\{a_i\}_{i=1}^k$, A applies a random permutation π . The i -th element becomes $\llbracket a'_i \rrbracket = \llbracket a_{\pi(i)} \rrbracket$ instead of $\llbracket a_i \rrbracket$.

Now, A and B compare the first two values $\llbracket a'_1 \rrbracket$ and $\llbracket a'_2 \rrbracket$ using the comparison protocol from row 4 of Table 2. B learns the index, m , of the larger value, and tells A to compare $\llbracket a'_m \rrbracket$ to $\llbracket a'_3 \rrbracket$ next. After iterating in this manner through all the k values, B determines the index m of the largest value. A can then compute $\pi^{-1}(m)$ which represents the argmax in the original, unpermuted order.

Since A applied a random permutation π , B does not learn the ordering of the values. The problem, though, is that A learns this ordering because, at every iteration, A knows the value of m up to that step and π . One way to fix this problem is for B to compare every pair of inputs from A , but this would result in a quadratic number of comparisons, which is too slow.

Instead, our protocol preserves the linear number of comparisons from above. The idea is that, at each iteration, once B determines which is the maximum of the two values compared, B should *randomize* the encryption of this maximum in such a way that A cannot link this value to one of the values compared. B uses the Refresh procedure for the randomization of Paillier ciphertexts. In the case where the “refresher” knows the secret key, this can be seen as a decryption followed by a re-encryption. If not, it can be seen as a multiplication by an encryption of 0.

A difficulty is that, to randomize the encryption of the maximum $\llbracket a'_m \rrbracket$, B needs to get this encryption – however, B must not receive this encryption because B has the key SK_P to decrypt it, which violates privacy. Instead, the idea is for A itself to add noise r_i and s_i to $\llbracket a'_m \rrbracket$, so decryption at B yields random values, then B refreshes the ciphertext, and then A removes the randomness r_i and s_i it added.

In the end, our protocol performs $k - 1$ encrypted comparisons of l bits integers and $7(k - 1)$ homomorphic operations (refreshes, multiplications and subtractions). In terms of round trips, we add $k - 1$ roundtrips to the comparison protocol, one roundtrip per loop iteration.

Proposition 4.1. *Protocol 1 is correct and secure in the honest-but-curious model.*

Proof intuition. The correctness property is straightforward. Let’s argue security. A does not learn intermediary results in the computation because of the security of the comparison protocol and because she gets a refreshed ciphertext from B which A cannot couple to a previously seen ciphertext. B does learn the result of each comparison – however, since A applied a random permutation before the comparison, B learns no useful information. See Appendix C for a complete proof.

Protocol 1 argmax over encrypted data

Input A: k encrypted integers $(\llbracket a_1 \rrbracket, \dots, \llbracket a_k \rrbracket)$, the bit length l of the a_i , and public keys PK_{QR} and PK_P

Input B: Secret keys SK_P and SK_{QR} , the bit length l

Output A: $\text{argmax}_i a_i$

```
1: A: chooses a random permutation  $\pi$  over  $\{1, \dots, k\}$ 
2: A:  $\llbracket \max \rrbracket \leftarrow \llbracket a_{\pi(1)} \rrbracket$ 
3: B:  $m \leftarrow 1$ 
4: for  $i = 2$  to  $k$  do
5:   Using the comparison protocol (Sec. 4.1.3), B gets the bit  $b_i = (\max \leq a_{\pi(i)})$ 
6:   A picks two random integers  $r_i, s_i \leftarrow (0, 2^{\lambda+l}) \cap \mathbb{Z}$ 
7:   A:  $\llbracket m'_i \rrbracket \leftarrow \llbracket \max \rrbracket \cdot \llbracket r_i \rrbracket$   $\triangleright m'_i = \max + r_i$ 
8:   A:  $\llbracket a'_i \rrbracket \leftarrow \llbracket a_{\pi(i)} \rrbracket \cdot \llbracket s_i \rrbracket$   $\triangleright a'_i = a_{\pi(i)} + s_i$ 
9:   A sends  $\llbracket m'_i \rrbracket$  and  $\llbracket a'_i \rrbracket$  to B
10:  if  $b_i$  is true then
11:    B:  $m \leftarrow i$ 
12:    B:  $\llbracket v_i \rrbracket \leftarrow \text{Refresh}(\llbracket a'_i \rrbracket)$   $\triangleright v_i = a'_i$ 
13:  else
14:    B:  $\llbracket v_i \rrbracket \leftarrow \text{Refresh}(\llbracket m'_i \rrbracket)$   $\triangleright v_i = m'_i$ 
15:  end if
16:  B sends to A  $\llbracket v_i \rrbracket$ 
17:  B sends to A  $\llbracket b_i \rrbracket$ 
18:  A:  $\llbracket \max \rrbracket \leftarrow \llbracket v_i \rrbracket \cdot (g^{-1} \cdot \llbracket b_i \rrbracket)^{r_i} \cdot \llbracket b_i \rrbracket^{-s_i}$ 
19:  $\triangleright \max = v_i + (b_i - 1) \cdot r_i - b_i \cdot t_i$ 
20: end for
21: B sends  $m$  to A
22: A outputs  $\pi^{-1}(m)$ 
```

4.3 Changing the encryption scheme

To enable us to compose various building blocks, we developed a protocol for converting ciphertexts from one encryption scheme to another while maintaining the underlying plaintexts. We first present a protocol that switches between two encryption schemes with the *same* plaintext size (such as QR and FHE over bits), and then present a different protocol for switching from QR to Paillier.

Concretely, consider two additively homomorphic encryption schemes E_1 and E_2 , both semantically secure with the same plaintext space M . Let $\llbracket \cdot \rrbracket_1$ be an encryption using E_1 and $\llbracket \cdot \rrbracket_2$ an encryption using E_2 . Consider that party B has the secret keys SK_1 and SK_2 for both schemes and A has the corresponding public keys PK_1 and PK_2 . Party A also has a value encrypted with PK_1 , $\llbracket c \rrbracket_1$. Our protocol, protocol 2, enables A to obtain an encryption of c under E_2 , $\llbracket c \rrbracket_2$ without revealing anything to B about c .

Protocol intuition. The idea is for A to add a random noise r to the ciphertext using the homomorphic property of E_1 . Then B decrypts the resulting value with E_1 (obtaining $x + r \in M$) and encrypts it with E_2 , sends the result to A which removes the randomness r using the homomorphic property of E_2 . Even though B was able to decrypt $\llbracket c' \rrbracket_1$, B obtains $x + r \in M$ which hides x in an information-theoretic way (it is a one-time pad).

Note that, for some schemes, the plaintext space M depends on the secret keys. In this case, we must be sure that party A can still choose uniformly elements of M without knowing it. For example, for Paillier, $M = \mathbb{Z}_N^* \simeq \mathbb{Z}_p^* \times \mathbb{Z}_q^*$ where p and q are the private primes. However, in this case, A can sample noise in \mathbb{Z}_N that will not be in \mathbb{Z}_N^* with negligible probability $(1 - \frac{1}{p})(1 - \frac{1}{q}) \approx 1 - \frac{2}{\sqrt{N}}$ (remember N is large – 1024 bits in our instantiation).

Proposition 4.2. *Protocol 2 is secure in the honest-but-curious model.*

In our classifiers, we use this protocol for $M = \{0, 1\}$ and the encryption schemes are QR (for E_1) and an FHE scheme over bits (for E_2). In some cases, we might also want to switch from QR to Paillier (e.g. reuse the encrypted

Protocol 2 Changing the encryption scheme

Input A: $\llbracket c \rrbracket_1$ and public keys PK_1 and PK_2

Input B: Secret keys SK_1 and SK_2

Output A: $\llbracket c \rrbracket_2$

- 1: A uniformly picks $r \leftarrow M$
 - 2: A sends $\llbracket c' \rrbracket_1 \leftarrow \llbracket c \rrbracket_1 \cdot \llbracket r \rrbracket_1$ to B
 - 3: B decrypts c' and re-encrypts with E_2
 - 4: B sends $\llbracket c' \rrbracket_2$ to A
 - 5: A: $\llbracket c \rrbracket_2 = \llbracket c' \rrbracket_2 \cdot \llbracket r \rrbracket_2^{-1}$
 - 6: A outputs $\llbracket c \rrbracket_2$
-

result of a comparison in a homomorphic computation), which has a different message space. Note that we can *simulate* the homomorphic XOR operation and a message space $M = \{0, 1\}$ with Paillier: we can easily compute the encryption of $b_1 \oplus b_2$ under Paillier when at most one of the b_i is encrypted (which we explain in the next subsection). This is the case in our setting because party A has the randomness r in the clear.

4.3.1 XOR with Paillier.

Suppose a party gets the bit b_1 encrypted under Paillier's encryption scheme, and that this party only has the public key. This party knows the bit b_2 in the clear and wants to compute the encryption of $\llbracket b_1 \oplus b_2 \rrbracket$.

To do so, we just have to notice that

$$b_1 \oplus b_2 = \begin{cases} b_1 & \text{if } b_2 = 0 \\ 1 - b_1 & \text{if } b_2 = 1 \end{cases}$$

Hence, it is very easy to compute an encryption of $b_1 \oplus b_2$ if we know the modulus N and the generator g (cf. Paillier's scheme construction):

$$\llbracket b_1 \oplus b_2 \rrbracket = \begin{cases} \llbracket b_1 \rrbracket & \text{if } b_2 = 0 \\ g^{\llbracket b_1 \rrbracket - 1} \bmod N^2 & \text{if } b_2 = 1 \end{cases}$$

If we want to unveil the result to an adversary who knows the original encryption of b_1 (but not the secret key), we have to refresh the result of the previous function to ensure semantic security.

4.4 Computing dot products

For completeness, we include a straightforward algorithm for computing dot products of two vectors, which relies on Paillier's homomorphic property.

Protocol 3 Private dot product

Input A: $x = (x_1, \dots, x_d) \in \mathbb{Z}^d$, public key PK_P

Input B: $y = (y_1, \dots, y_d) \in \mathbb{Z}^d$, secret key SK_P

Output A: $\llbracket \langle x, y \rangle \rrbracket$

- 1: B encrypts y_1, \dots, y_d and sends the encryptions $\llbracket y_i \rrbracket$ to A
 - 2: A computes $\llbracket v \rrbracket = \prod_i \llbracket y_i \rrbracket^{x_i} \bmod N^2$
 - 3: A re-randomizes and outputs $\llbracket v \rrbracket$
-

$$\triangleright v = \sum y_i x_i$$

Proposition 4.3. *Protocol 3 is secure in the honest-but-curious model.*

4.5 Dealing with floating point numbers

Although all our protocols manipulate integers, classifiers usually use floating point numbers. Hence, when developing classifiers with our protocol library, we must adapt our protocols accordingly.

Fortunately, most of the operations involved are either additions or multiplications. As a consequence, a simple solution is to multiply each floating point value by a constant K (e.g. $K = 2^{52}$ for IEEE 754 doubles) and thus support finite precision. We must also consider the bit length for the comparisons. We show an example of a full analysis in Section 6 for the Naïve Bayes classifier.

5 Private hyperplane decision

Recall from Section 3.1 that this classifier computes

$$k^* = \operatorname{argmax}_{i \in [k]} \langle w_i, x \rangle.$$

Now that we constructed our library of building blocks, it is straightforward to implement this classifier securely: the client computes the encryption of $\llbracket \langle w_i, x \rangle \rrbracket$ for all $i \in [k]$ using the dot product protocol and then applies the argmax protocol (Protocol 1) to the encrypted dot products.

Protocol 4 Private hyperplane decision

Client's (C) Input: $x = (x_1, \dots, x_d) \in \mathbb{Z}^d$, public keys PK_P and PK_{QR}

Server's (S) Input: $\{w_i\}_{i=1}^k$ where $\forall i \in [k], w_i \in \mathbb{Z}^n$, secret keys SK_P and SK_{QR}

Client's Output: $\operatorname{argmax}_{i \in [k]} \langle w_i, x \rangle$

- 1: **for** $i = 1$ **to** k **do**
 - 2: C and S run Protocol 3 for private dot product where C is party A with input x and S is party B with input w_i .
 - 3: C gets $\llbracket v_i \rrbracket$ the result of the protocol.

$\triangleright v_i \leftarrow \langle x, w_i \rangle$
 - 4: **end for**
 - 5: C and S run Protocol 1 for argmax where C is the A, and S the B, and $\llbracket v_1 \rrbracket, \dots, \llbracket v_k \rrbracket$ the input ciphertexts. C gets the result i_0 of the protocol.

$\triangleright i_0 \leftarrow \operatorname{argmax}_{i \in [k]} v_i$
 - 6: C outputs i_0
-

Proposition 5.1. *Protocol 4 is secure in the honest-but-curious model.*

6 Secure Naïve Bayes classifier

Section 3.1 describes the Naïve Bayes classifier. The goal is for the client to learn k^* without learning anything about the probabilities that constitute the model, and the server should learn nothing about x . Recall that the features values domain is discrete and finite.

As is typically done for numerical stability reasons, we work with the logarithm of the probability distributions:

$$\begin{aligned} k^* &= \operatorname{argmax}_{i \in [k]} \log p(C = c_i | X = x) \\ &= \operatorname{argmax}_{i \in [k]} \left\{ \log p(C = c_i) + \sum_{j=1}^d \log p(X_j = x_j | C = c_i) \right\} \end{aligned} \quad (2)$$

6.1 Preparing the model

Since the Paillier encryption scheme works with integers, we convert each log of a probability from above to an integer by multiplying it with a large number K (recall that the plaintext space of Paillier is large $\approx 2^{1024}$ thus allowing for a large K), thus still maintaining high accuracy. The issues due to using integers for bayesian classification have been previously studied in [TRMP12], even though their setting was even more restricting than ours. However, they use a similar idea to ours: *shifting* the probabilities logarithms and use fixed point representation.

As the only operations used in the classification step are additions and comparisons (cf. Equation (2)), we can just multiply the conditional probabilities $p(x_j|c_i)$ by a constant K so to get integers everywhere, while keeping the same classification result.

For example, if we are able to compute the conditional probabilities using IEEE 754 double precision floating point numbers, with 52 bits of precision, then we can represent every probability p as

$$p = m \cdot 2^e$$

where m binary representation is $(m)_2 = 1.d$ and d is a 52 bits integer. Hence we have $1 \leq m < 2$ and we can rewrite m as

$$m = \frac{m'}{2^{52}} \text{ with } m' \in \mathbb{N} \cap [2^{52}, 2^{53})$$

We are using this representation to find a constant K such that $K \cdot v_i \in \mathbb{N}$ for all i . As seen before, we can write the v_i 's as

$$v_i = m'_i \cdot 2^{e_i - 52}$$

Let $e^* = \min_i e_i$, and $\delta_i = e_i - e^* \geq 0$. Then,

$$v_i = m'_i \cdot 2^{\delta_i} \cdot 2^{e^* - 52}$$

So let $K = 2^{52 - e^*}$. We have $K \cdot v_i = m'_i \cdot 2^{\delta_i} \in \mathbb{N}$. An important thing to notice is that the v_i 's can be very large integers (due to δ_i), and this might cause overflows errors. However, remember that we are doing all this to store logarithms of probabilities in Paillier cyphertexts, and as Paillier plaintext space is *very* large (more than 1024 bits in our setting) and δ_i 's remain small³. Also notice that this *shifting* procedure can be done without any loss of precision as we can directly work with the bit representation of the floating points numbers.

Finally, we must also ensure that we do not overflow Paillier's message space when doing all the operations (homomorphic additions, comparisons, ...). If – as before – d is the number of features, the maximum number of bits when doing the computations will be $l_{max} = d + 1 + (52 + \delta^*)$ where $\delta^* = \max \delta_i$: we have to add the probabilities for the d features and the probability of the class label (the $d + 1$ term), and each probability is encoded using $(52 + \delta^*)$ bits. Hence, the value l used for the comparison protocols must be chosen larger than l_{max} .

Hence, we must ensure that $\log_2 N > l_{max} + 1 + \lambda$ where λ is the security parameter and N is the modulus for Paillier's cryptosystem plaintext space (cf. Section 4.1.2). This condition is easily fulfilled as, for a good level of security, we have to take $\log_2 N \geq 1024$ and we usually take $\lambda \approx 100$.

Let D_j be the domain of possible values of x_j (the j -th attribute of the feature vector x). The server prepares $kd + 1$ tables as part of the model, where K is computed as described just before:

- One table for the priors on the classes P : $P(i) = \lceil K \log p(C = c_i) \rceil$.
- One table per feature j per class i , $T_{i,j}$: $T_{i,j}(v) \approx \lceil K \log p(X_j = v | C = c_i) \rceil$, for all $v \in D_j$.

The tables remain small: P has one entry by category *i.e.* k entries total, and T has one entry by category and feature value *i.e.* $k \cdot D$ entries where $D = \sum |D_j|$. In our examples, this represents less than 3600 entries. Moreover, this preparation step can be done once and for all at server startup, and is hence amortized.

³If the biggest δ_i is 10, the ratio between the smallest and the biggest probability is of order $2^{2^{10}} = 2^{1024} \dots$

6.2 Protocol

Let us begin with some intuition. The server encrypts each entry in these tables with Paillier and gives the resulting encryption (the encrypted model) to the client. For every class c_i , the client uses Paillier's additive homomorphism to compute $\llbracket p_i \rrbracket = \llbracket P(i) \rrbracket \prod_{j=1}^d \llbracket T_{i,j}(x_j) \rrbracket$. Finally, the client runs the argmax protocol, Protocol 1, to get $\text{argmax } p_i$. For completeness, the protocol is shown in Protocol 5.

Protocol 5 Naïve Bayes Classifier

Client's (C) Input: $x = (x_1, \dots, x_d) \in \mathbb{Z}^d$, public key PK_P , secret key SK_{QR}

Server's (S) Input: The secret key SK_P , public key PK_{QR} and probability tables $\{\log p(C = c_i)\}_{1 \leq i \leq k}$ and $\{\{\log p(X_j = v | C = c_i)\}_{v \in D_j}\}_{1 \leq j \leq d, 1 \leq i \leq k}$

Client's Output: i_0 such that $p(x, c_{i_0})$ is maximum

- 1: The server prepares the tables P and $\{T_{i,j}\}_{1 \leq i \leq k, 1 \leq j \leq d}$ and encrypts their entries using Paillier.
 - 2: The server sends $\llbracket P \rrbracket$ and $\{\llbracket T_{i,j} \rrbracket\}_{i,j}$ to the client.
 - 3: For all $1 \leq i \leq k$, the client computes $\llbracket p_i \rrbracket = \llbracket P(i) \rrbracket \prod_{j=1}^d \llbracket T_{i,j}(x_j) \rrbracket$.
 - 4: The client runs the argmax protocol (Protocol 1) with the server and gets $i_0 = \text{argmax}_i p_i$
 - 5: C outputs i_0
-

Proposition 6.1. *Protocol 5 is secure in the honest-but-curious model.*

Proof intuition. Given the security property of the argmax protocol, Protocol 1, and the semantic security of the Paillier cryptosystem, the security of this classifier follows trivially, by invoking a modular composition theorem.

Efficiency. Note that the tables P and $\{T_{i,j}\}_{1 \leq i \leq k, 1 \leq j \leq d}$ can be prepared in advance. Hence the cost of constructing the tables can be amortized over many uses. To compute the encrypted probabilities p_i 's, the client runs d homomorphic operations (here multiplications) for each i , hence doing kd modular multiplications. Then the parties run a single argmax protocol *i.e.* $k - 1$ comparisons and $O(k)$ homomorphic operations. Thus, compared to non-encrypted computation, the overhead comes only from the use of homomorphic encryption operations instead of plaintext operations. Regarding the number of round trips, these are due to the argmax protocol: $k - 1$ runs of the comparison protocol and $k - 1$ additional roundtrips.

7 Private decision trees

A private decision tree classifier allows the server to traverse a binary decision tree using the client's input x such that the server does not learn the input x , and the client does not learn the structure of the tree and the thresholds at each node. A challenge is that, in particular, the client should not learn the path in the tree that corresponds to x – the position of the path in the tree and the length of the path leaks information about the model. The outcome of the classification does not necessarily leak the path in the tree

The idea is to express the decision tree as a polynomial P whose output is the result of the classification, the class predicted for x . Then, the server and the client privately compute inputs to this polynomial based on x and the thresholds w_i . Finally, the server evaluates the polynomial P privately.

7.1 Polynomial form of a decision tree

Consider that each node of the tree has a boolean variable associated to it. The value of the boolean at a node is 1 if, on input x , one should follow the right branch, and 0 otherwise. For example, denote the boolean variable at the root of the tree by b_1 . The value of b_1 is 1 if $x_1 \leq w_1$ (recall Figure 2), and 0 otherwise.

We construct a polynomial P that, on input all these boolean variables and the value of each class at a leaf node, outputs the class predicted for x . The idea is that P is a sum of terms, where each term (say t) corresponds

to a path in the tree from root to a leaf node (say c). A term t evaluates to c iff x is classified along that path in T , else it evaluates to zero. Hence, the term corresponding to a path in the tree is naturally the multiplication of the boolean variables on that path and the class at the leaf node. For example, for the tree in Figure 3, P is $P(b_1, b_2, b_3, b_4, c_1, \dots, c_5) = b_1(b_3 \cdot (b_4 \cdot c_5 + (1 - b_4) \cdot c_4) + (1 - b_3) \cdot c_3) + (1 - b_1)(b_2 \cdot c_2 + (1 - b_2) \cdot c_1)$.

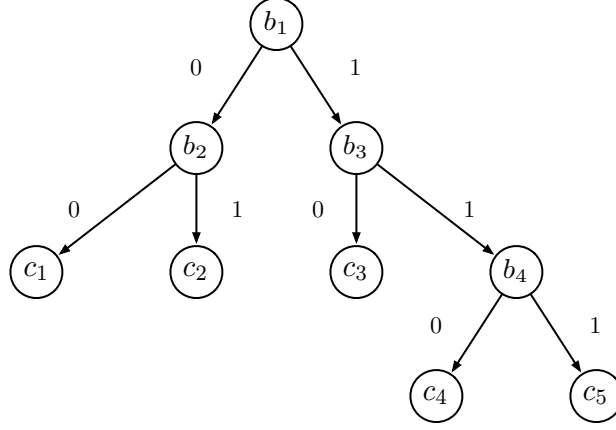
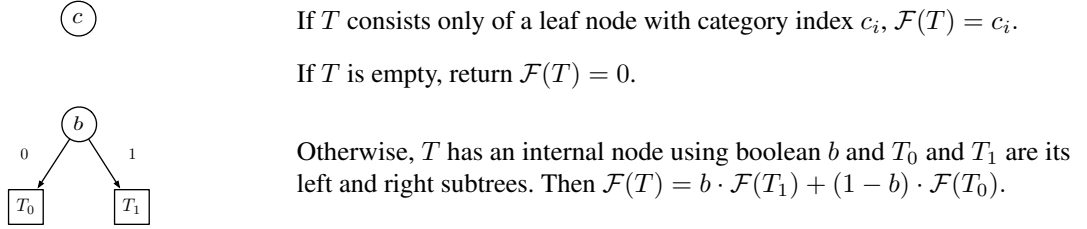


Figure 3: Decision tree with booleans

We now present \mathcal{F} , a recursive procedure for constructing P given a binary decision tree T :



7.2 Private evaluation of a polynomial

Let us first explain how to compute the values of the boolean variables securely. Let n be the number of nodes in the tree and n_{leaves} be the number of leaves in the tree. These values must remain unknown to the server because they leak information about x : they are the result of the intermediate computations of the classification criterion. For each boolean variable b_i , the server and the client engage in the comparison protocol to compare w_i and the corresponding attribute of x . As a result, the server obtains $[b_i]$ for $i \in 1 \dots n$; the server then changes the encryption of these values to FHE using Protocol 2, thus obtaining $\llbracket b_i \rrbracket$.

The server evaluates P on $(\llbracket b_1 \rrbracket, \dots, \llbracket b_n \rrbracket)$ using the homomorphic properties of FHE. In most cases, FHE evaluation is very slow, but we succeed to make it efficient through a combination of techniques we now discuss. To understand these techniques, recall that a typical FHE evaluation happens over a circuit whose gates are modular addition and multiplication. The performance of FHE depends a lot on the depth of multiplications in this circuit.

First, we use a *leveled* FHE scheme: a scheme that supports only an a priori fixed multiplicative depth instead of an arbitrary such depth. As long as this depth is small, such a scheme is much faster than a *full* FHE scheme.

Second, we ensure that the multiplicative depth is very small using a tree-based evaluation. If h_{max} is the maximum height of the decision tree, then P has a term $a_1 \cdot \dots \cdot a_{h_{\text{max}}}$. If we evaluate this term naively with FHE, we multiply these values sequentially. This yields a multiplicative depth of h_{max} , which makes FHE slow for common h_{max} values. Instead, we construct a binary tree over these values and multiply them in pairs based on the structure of this tree. This results in a multiplicative depth of $\log_2 h_{\text{max}}$ (e.g., 4), which makes FHE evaluation significantly more efficient.

Finally, we use \mathbb{F}_2 as the plaintext space and SIMD slots for parallelism. FHE schemes are significantly faster when the values encrypted are bits (namely, in \mathbb{F}_2); however, P contains classes (e.g., c_1) which are usually more than a bit in length. To enable computing P over F_2 , we represent each class in binary. Let $l = \lceil \log_2 k \rceil$ (k is the number of classes) be the number of bits needed to represent a class. We evaluate P l times, once for each of the l bits of a class. Concretely, the j -th evaluation of P takes as input b_1, \dots, b_n and for each leaf node c_i , its j -th bit c_{ij} . The result is $P(b_1, \dots, b_n, c_{1j}, c_{2j}, \dots, c_{n\text{leaves}j})$, which represents the j -th bit of the outcome class. Hence, we need to run the FHE evaluation l times.

To avoid this factor of l , the idea is to use a nice feature of FHE called *SIMD slots* (as described in [SV11]): these allow encrypting multiple bits in a single ciphertext such that any operation applied to the ciphertext gets applied in parallel to each of the bits. Hence, for each class c_j , the server creates an FHE ciphertext $\llbracket c_{j0}, \dots, c_{jl-1} \rrbracket$. For each node b_i , it creates an FHE ciphertext $\llbracket b_i, \dots, b_i \rrbracket$ by simply repeating the b_i value in each slot. Then, the server runs one FHE evaluation of P over all these ciphertexts and obtains $\llbracket c_{o0}, \dots, c_{ol-1} \rrbracket$ where c_o is the outcome class. Hence, instead of l FHE evaluations, the server runs the evaluation only once. This results in a performance improvement of $\log k$, a factor of 2 and more in our experiments. We were able to apply SIMD slots parallelism due to the fortunate fact that the same polynomial P had to be computed for each slot.

Finally, evaluating the decision tree is done using $2n$ FHE multiplications and $2n$ FHE additions where n is the number of criteria. The evaluation circuit has multiplication depth $\lceil \log_2(n) + 1 \rceil$.

7.3 Formal description

Protocol 6 describes the resulting protocol.

Protocol 6 Decision Tree Classifier

Client's (C) Input: $x = (x_1, \dots, x_n) \in \mathbb{Z}^n$, secret keys $\text{SK}_{QR}, \text{SK}_{FHE}$

Server's (S) Input: The public keys $\text{PK}_{QR}, \text{PK}_{FHE}$, the model as a decision tree, including the n thresholds $\{w_i\}_{i=1}^n$.

Client's Output: The value of the leaf of the decision tree associated with the inputs b_1, \dots, b_n .

- 1: S produces an n -variate polynomial P as described in section 7.1.
 - 2: S and C interact in the comparison protocol, so that S obtains $[b_i]$ for $i \in [1 \dots n]$ by comparing w_i to the corresponding attribute of x .
 - 3: Using Protocol 2, S changes the encryption from QR to FHE and obtains $\llbracket b_1 \rrbracket, \dots, \llbracket b_n \rrbracket$.
 - 4: To evaluate P , S encrypts the bits of each category c_i using FHE and SIMD slots, obtaining $\llbracket c_{i1}, \dots, c_{il} \rrbracket$. S uses SIMD slots to compute homomorphically $\llbracket P(b_1, \dots, b_n, c_{10}, \dots, c_{n\text{leaves}0}), \dots, P(b_1, \dots, b_n, c_{1l-1}, \dots, c_{n\text{leaves}l-1}) \rrbracket$. It rerandomizes the resulting ciphertext using FHE's rerandomization function, and sends the result to the client.
 - 5: C decrypts the result as the bit vector (v_0, \dots, v_{l-1}) and outputs $\sum_{i=0}^{l-1} v_i \cdot 2^i$.
-

Proposition 7.1. *Protocol 6 is secure in the honest-but-curious model.*

Proof intuition. The proof is in Appendix C, but we give some intuition here. During the comparison protocol, the server only learns encrypted bits, so it learns nothing about x . During FHE evaluation, it similarly learns nothing about the input due to the security of FHE. The client does not learn the structure of the tree because the server performs the evaluation of the polynomial. Similarly, the client does not learn the bits at the nodes in the tree because of the security of the comparison protocol.

The interactions between the client and the server are due to the comparisons almost exclusively: the decision tree evaluation does not need any interaction but sending the encrypted result of the evaluation.

```

bool Linear_Classifier_Client::run()
{
    exchange_keys();

    // values_ is a vector of integers
    // compute the dot product
    mpz_class v = compute_dot_product(values_);
    mpz_class w = 1; // encryption of 0

    // compare the dot product with 0
    return enc_comparison(v, w, bit_size_, false);
}

void Linear_Classifier_Server_session::
    run_session()
{
    exchange_keys();

    // enc_model_ is the encrypted model vector
    // compute the dot product
    help_compute_dot_product(enc_model_, true);

    // help the client to get
    // the sign of the dot product
    help_enc_comparison(bit_size_, false);
}

```

Figure 4: Implementation example: a linear classifier

Bit size	A Computation	B Computation	Total Time	Communication	Interactions
10	14.11 ms	8.39 ms	105.5 ms	4.60 kB	3
20	18.29 ms	14.1 ms	117.5 ms	8.82 kB	3
32	22.9 ms	18.8 ms	122.6 ms	13.89 kB	3
64	34.7 ms	32.6 ms	134.5 ms	27.38 kB	3

Table 3: Comparison with unencrypted input protocols evaluation.

8 Combining classifiers with AdaBoost

AdaBoost is a technique introduced in [FS97]. The idea is to combine a set of *weak* classifiers $h_i(x) : \mathbb{R}^d \mapsto \{-1, +1\}$ to obtain a better classifier. The AdaBoost algorithm chooses t scalars $\{\alpha_i\}_{i=1}^t$ and constructs a strong classifier as:

$$H(x) = \text{sign} \left(\sum_{i=1}^t \alpha_i h_i(x) \right)$$

If each of the $h_i(\cdot)$'s is an instance of a classifier supported by our protocols, then given the scalars α_i , we can easily and securely evaluate $H(x)$ by simply composing our building blocks. First, we run the secure protocols for each of h_i , except that the server keeps the intermediate result, the outcome of $h_i(x)$, encrypted using one of our comparison protocols (Rows 2 or 4 of Table 2). Second, if necessary, we convert them to Paillier's encryption scheme with Protocol 2, and combine these intermediate results using Paillier's additive homomorphic property as in the dot product protocol Protocol 3. Finally, we run the comparison over encrypted data algorithm to compare the result so far with zero, so that the client gets the final result.

9 Implementation

We have implemented the protocols and the classifiers in C++ using GMP⁴, Boost, Google's Protocol Buffers⁵, and HELib [Hal13] for the FHE implementation.

The code is written in a modular way: all the elementary protocols defined in Section 4 can be used as black boxes with minimal developer effort. Thus, writing secure classifiers comes down to invoking the right API calls to the protocols. For example, for the linear classifier, the client simply calls a key exchange protocol to setup the various keys, followed by the dot product protocol, and then the comparison of encrypted data protocol to output the result, as shown in Figure 4.

⁴<http://gmplib.org/>

⁵<https://code.google.com/p/protobuf/>

Protocol	Bit size	Computation		Total Time	Communication	Interactions
		Party A	Party B			
Comparison	64	45.34 ms	43.78 ms	190.9 ms	27.91 kB	6
Reversed Comp.	64	48.78 ms	42.49 ms	195.7 ms	27.91 kB	6

Table 4: Comparison with encrypted input protocols evaluation.

Party A Computation	Party B Computation	Total Time	Communication	Interactions
30.80 ms	255.3 ms	360.7 ms	420.1 kB	2

Table 5: Change encryption scheme protocol evaluation.

10 Evaluation

To evaluate our work, we answer the following questions: (i) can our building blocks be used to construct other classifiers in a modular way (Section 10.1), (ii) what is the performance overhead of our building blocks (Section 10.3), and (iii) what is the performance overhead of our classifiers (Section 10.4)?

10.1 Using our building blocks library

Here we demonstrate that our building blocks library can be used to build other classifiers modularly and that it is a useful contribution by itself. We will construct a multiplexer and a face detector. A face detection algorithm over encrypted data already exists [AB06, AB07], so our construction here is not the first such construction, but it serves as a proof of functionality for our library.

10.1.1 Building a multiplexer classifier

A multiplexer is the following generalized comparison function:

$$f_{\alpha,\beta}(a,b) = \begin{cases} \alpha & \text{if } a > b \\ \beta & \text{otherwise} \end{cases}$$

We can express $f_{\alpha,\beta}$ as a linear combination of the bit $d = (a \leq b)$:

$$f_{\alpha,\beta}(d) = d \cdot \beta + (1 - d) \cdot \alpha = \alpha + d \cdot (\beta - \alpha).$$

To implement this classifier privately, we compute $\llbracket d \rrbracket$ by comparing a and b , keeping the result encrypted with QR, and then changing the encryption scheme (cf. Section 4.3) to Paillier.

Then, using Paillier’s homomorphism and knowledge of α and β , we can compute an encryption of $f_{\alpha,\beta}(d)$:

$$\llbracket f_{\alpha,\beta}(d) \rrbracket = \llbracket \alpha \rrbracket \cdot \llbracket d \rrbracket^{\beta - \alpha}.$$

10.1.2 Viola and Jones face detection

The Viola and Jones face detection algorithm [VJ01] is a particular case of an AdaBoost classifier. Denote by X an image represented as an integer vector and x a particular detection window (a subset of X ’s coefficients). The *strong* classifier H for this particular detection window is

$$H(x) = \text{sign} \left(\sum_{i=1}^t \alpha_i h_i(x) \right)$$

where the h_t are weak classifiers of the form $h_i(x) = \text{sign}(\langle x, y_i \rangle - \theta_i)$.

Data set	Model size	Computation		Time per protocol		Total running time	Comm.	Interactions
		Client	Server	Compare	Dot product			
Breast cancer (2)	30	46.4 ms	43.8 ms	194 ms	9.67 ms	204 ms	35.84 kB	7
Credit (3)	47	55.5 ms	43.8 ms	194 ms	23.6 ms	217 ms	40.19 kB	7

(a) Linear Classifier. Time per protocol includes communication.

Data set	Specs.		Computation		Time per protocol		Total running time	Comm.	Interactions
	C	F	Client	Server	Prob. Comp.	Argmax			
Breast Cancer (1)	2	9	150 ms	104 ms	82.9 ms	396 ms	479 ms	72.47 kB	14
Nursery (5)	5	9	537 ms	368 ms	82.8 ms	1332 ms	1415 ms	150.7 kB	42
Audiology (4)	24	70	1652 ms	1664 ms	431 ms	3379 ms	3810 ms	1911 kB	166

(b) Naïve Bayes Classifier. C is the number of classes and F is the number of features. The Prob. Comp. column corresponds to the computation of the probabilities $p(c_i|x)$ (cf. Section 6). Time per protocol includes communication.

Data set	Tree Specs.		Computation		Time per protocol		FHE		Comm.	Interactions
	N	D	Client	Server	Compare	ES Change	Eval.	Decrypt		
Nursery (5)	4	4	1579 ms	798 ms	446 ms	1639 ms	239 ms	33.51 ms	2639 kB	30
ECG (6)	6	4	2297 ms	1723 ms	1410 ms	7406 ms	899 ms	35.1 ms	3555 kB	44

(c) Decision Tree Classifier. ES change indicates the time to run the protocol for changing encryption schemes. N is the number of nodes of the tree and D is its depth. Time per protocol includes communication.

Table 6: Classifiers evaluation.

In our setting, Alice owns the image and Bob the classifier (e.g. the vectors $\{y_i\}$ and the scalars $\{\theta_i\}$ and $\{\alpha_i\}$). Neither of them wants to disclose their input to the other party. Thanks to our building blocks, Alice can run Bob’s classifier on her image without her learning anything about the parameters and Bob learning any information about her image.

The weak classifiers can be seen as multiplexers; with the above notation, we have $h_t(x) = f_{1,-1}(\langle x, y_t \rangle - \theta_t)$.

Using the elements of Section 10.1.1, we can easily compute the encrypted evaluation of every one of these weak classifiers under Paillier, and then, as described in Section 8, compute the encryption of $H(x)$.

10.2 Performance evaluation setup

Our performance evaluations were run using two desktop computers each with identical configuration: two Intel Core i7 (64 bit) processors for a total 4 cores running at 2.66 GHz and 8 GB RAM. Since the machines were on the same network, we inflated the roundtrip time for a packet to be 40 ms to mimic real network latency. We used 1024-bit cryptographic keys, and chose the statistical security parameter λ to be 100. When using HELib, we use 80 bits of security, which corresponds to a 1024-bit asymmetric key.

10.3 Building blocks performance

We examine performance in terms of computation time at the client and server, communication bandwidth, and also number of interactions (round trips). We can see that all these protocols are efficient, with a runtime on the order of milliseconds.

10.3.1 Comparison protocols

Comparison with unencrypted input. Table 3 gives the running time of the comparison protocol with unencrypted input for various input size.

Comparison with encrypted input. Table 4 presents the performance of the comparison with encrypted inputs protocols.

10.3.2 argmax

Figure 5 presents the running times and the communication overhead of the argmax of encrypted data protocol (*cf.* Section 4.2). The input integers were 64 bit integers.

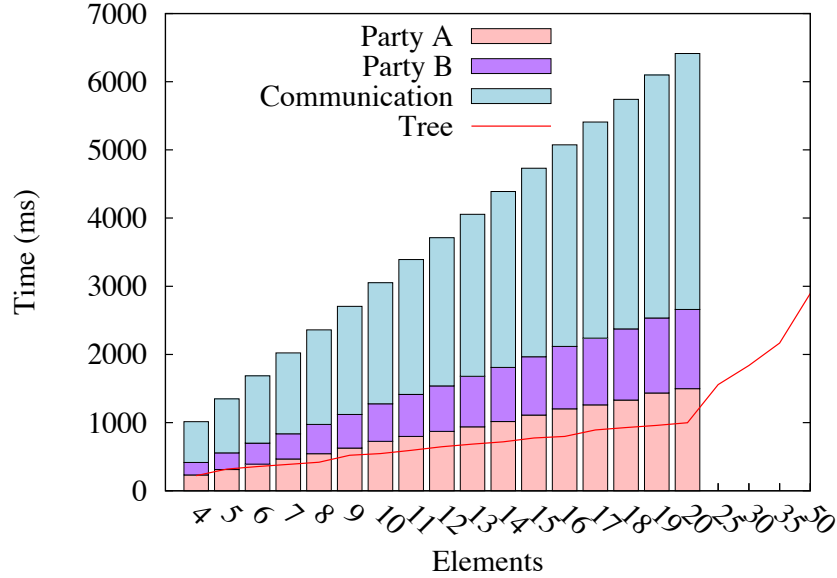


Figure 5: Argmax of encrypted data protocol evaluation. The bars represent the execution of the protocol when the comparisons are executed one after each other, linearly. The line represents the execution when comparisons are executed in parallel, tree-wise.

10.3.3 Consequences of the latency on performances

It is worth noticing that for most blocks, most of the running time is spend communicating: the network’s latency has a huge influence on the performances of the protocols (running time almost linear in the latency for some protocols). To improve the performances of a classifier implemented with our blocks, we might want to run several instances of some building blocks in parallel. This is actually what we did with the tree-based implementation of the argmax protocol, greatly improving the performances of the protocol (*cf.* Figure 5).

10.4 Classifier performance

Here we evaluate each of the classifiers described in Sections 5–7. The models are trained non-privately using `scikit-learn`⁶. We used the following datasets from the UCI machine learning repository [BL13]:

1. the Wisconsin Diagnostic Breast Cancer data set,
2. the Wisconsin Breast Cancer (Original) data set, a simplified version of the previous dataset,
3. Credit Approval data set,
4. Audiology (Standardized) data set,
5. Nursery data set, and

⁶<http://scikit-learn.org>

6. ECG (electrocardiogram) classification data from Barni *et al.* [BFK⁺09]

These data sets are scenarios when we want to ensure privacy of the server’s model and client’s input.

Based on the suitability of each classifier, we used data sets 2 and 3 to test the hyperplane decision classifier, sets 1, 4 and 5 for the Naïve Bayes classifier, and sets 5 and 6 for the decision tree classifier.

Table 6 shows the performance results. Our classifiers run in at most a few seconds, which we believe to be practical for sensitive applications. Note that even if the datasets become very large, the size of the model stays the same – the dataset size only affects the training phase which happens on unencrypted data before one uses our classifiers. Hence, the cost of our classification will be the same even for very large data sets.

For the decision tree classifier, we compared our construction to Barni *et al.* [BFK⁺09] on the ECG dataset (by turning their branching program into a decision tree). Their performance is 2609 ms⁷ for the client and 6260 ms for the server with communication cost of 112.2KB. Even though their evaluation does not consider the communication delays, we are still more than three times as fast for the server and faster for the client.

10.5 Comparison to generic two-party tools

A set of generic secure two- or multi-party computation tools have been developed, such as TASTY [HKS⁺10] and Fairplay [MNPS04, BDNP08]. These support general functions, which include our classifiers.

However, they are prohibitively slow for our specific setting. Our efficiency comes from specializing to classification functionality. To demonstrate their performance, we attempted to evaluate the Naïve Bayes classifier with these. We used FairplayMP to generate the circuit for this classifier and then TASTY to run the private computation on the circuit thus obtained. We tried to run the smallest Naïve Bayes instance, the Nursery dataset from our evaluation, which has only 3 possible values for each feature, but we ran out of memory during the circuit generation phase on a powerful machine with 256GB of RAM.

Hence, we had to reduce the classification problem to only 3 classes (versus 5). Then, the circuit generation took more than 2 hours with FairplayMP, and the time to run the classification with TASTY was 413196 msec (with no network delay), which is ≈ 500 times slower than our performance (on the non-reduced classification problem with 5 classes). Thus, our specialized protocols improve performance by orders of magnitude.

11 Conclusion

In this paper, we constructed three major privacy-preserving classifiers as well as provided a library of building blocks that enables constructing other classifiers. We demonstrated the efficiency of our classifiers and library on real datasets.

Acknowledgment

We thank Thijs Veugen, Thomas Schneider, and the anonymous reviewers for their helpful comments.

⁷In Barni *et al.* [BFK⁺09], the evaluation was run over two 3GHz computers directly connected via Gigabit Ethernet. We scaled the given results by $\frac{3}{2.3}$ to get a better comparison basis.

References

- [AB06] Shai Avidan and Moshe Butman. Blind vision. In *Computer Vision–ECCV 2006*, pages 1–13. 2006.
- [AB07] Shai Avidan and Moshe Butman. Efficient methods for privacy preserving face detection. In *Advances in Neural Information Processing Systems*, page 57, 2007.
- [AD01] Mikhail J Atallah and Wenliang Du. Secure multi-party computational geometry. In *Algorithms and Data Structures*, pages 165–179. 2001.
- [BDMN05] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In *Proceedings of the twenty-fourth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 128–138, 2005.
- [BDNP08] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: A system for secure multi-party computation. In *CCS*, pages 17–21, 2008.
- [BFK⁺09] Mauro Barni, Pierluigi Failla, Vladimir Kolesnikov, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. Secure evaluation of private linear branching programs with medical applications. In *Computer Security (ESORICS)*, pages 424–439. 2009.
- [BFL⁺09] Mauro Barni, Pierluigi Failla, Riccardo Lazzeretti, Annika Paus, A-R Sadeghi, Thomas Schneider, and Vladimir Kolesnikov. Efficient privacy-preserving classification of ecg signals. In *Information Forensics and Security, 2009. WIFS 2009. First IEEE International Workshop on*, pages 91–95, 2009.
- [BFL⁺11] Mauro Barni, Pierluigi Failla, Riccardo Lazzeretti, Ahmad-Reza Sadeghi, and Thomas Schneider. Privacy-preserving ECG classification with branching programs and neural networks. *IEEE Transactions on Information Forensics and Security (TIFS)*, 6(2):452–468, June 2011.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.
- [BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *IEEE SP*, pages 478–492, 2013.
- [BL13] K. Bache and M. Lichman. UCI machine learning repository, 2013.
- [BLN13] Joppe W. Bos, Kristin Lauter, and Michael Naehrig. Private predictive analysis on encrypted medical data. In *Microsoft Tech Report 200652*, 2013.
- [BN06] Christopher M Bishop and Nasser M Nasrabadi. Pattern recognition and machine learning. In *Journal of Electronic Imaging*, volume 1, 2006.
- [Can98] Ran Canetti. Security and composition of multi-party cryptographic protocols. *JOURNAL OF CRYPTOLOGY*, 13:2000, 1998.
- [CMS11] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12, 2011.
- [DGK07] Ivan Damgård, Martin Geisler, and Mikkel Krøigaard. Efficient and secure comparison for on-line auctions. In *Information Security and Privacy*, pages 416–430, 2007.
- [DGK09] Ivan Damgård, Martin Geisler, and Mikkel Kroigard. A correction to efficient and secure comparison for on-line auctions. 1(4):323–324, 2009.
- [DHC04] Wenliang Du, Yunghsiang S Han, and Shigang Chen. Privacy-preserving multivariate statistical analysis: Linear regression and classification. In *Proceedings of the 4th SIAM International Conference on Data Mining*, volume 233, 2004.

- [EFG⁺09] Zekeriya Erkin, Martin Franz, Jorge Guajardo, Stefan Katzenbeisser, Inald Lagendijk, and Tomas Toft. Privacy-preserving face recognition. In *Privacy Enhancing Technologies*, pages 235–253, 2009.
- [FS97] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, (1):119–139, 1997.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [GLLM04] Bart Goethals, Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. On private scalar product computation for privacy-preserving data mining. In *Information Security and Cryptology (ICISC)*, pages 104–120. 2004.
- [GLN12] Thore Graepel, Kristin Lauter, and Michael Naehrig. ML confidential: Machine learning on encrypted data. In *Information Security and Cryptology (ICISC)*, pages 1–21. 2012.
- [GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377. ACM, 1982.
- [GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *STOC*, pages 218–229, 1987.
- [Gol04] Oded Goldreich. *Foundations of Cryptography - Basic Applications*. Cambridge University Press, 2004.
- [Hal13] Shai Halevi. Helib - an implementation of homomorphic encryption. <https://github.com/shaih/HElib>, 2013.
- [HKS⁺10] Wilko Henecka, Stefan Kögl, Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Tasty: Tool for automating secure two-party computations. In *CCS*, pages 451–462, 2010.
- [IPS08] Yuval Ishai, Manoj Prabhakaran, and Amit Sahai. Founding cryptography on oblivious transfer – efficiently. In *Advances in Cryptology – CRYPTO 2008*, volume 5157, pages 572–591. 2008.
- [Kil05] Eike Kiltz. Unconditionally secure constant round multi-party computation for equality, comparison, bits and exponentiation. *IACR Cryptology ePrint Archive*, page 66, 2005.
- [KSS09] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. How to combine homomorphic encryption and garbled circuits - improved circuits and computing the minimum distance efficiently. In *1st International Workshop on Signal Processing in the EncryptEd Domain (SPEED’09)*, 2009.
- [KSS13] Vladimir Kolesnikov, Ahmad-Reza Sadeghi, and Thomas Schneider. A systematic approach to practically efficient general two-party secure function evaluation protocols and their modular design. In *Journal of Computer Security*, 2013.
- [LLM06] Sven Laur, Helger Lipmaa, and Taneli Mielikäinen. Cryptographically private support vector machines. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 618–624. ACM, 2006.
- [LP00] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Advances in Cryptology (CRYPTO)*, pages 36–54, 2000.
- [LP07] Yehuda Lindell and Benny Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. In *Advances in Cryptology (EUROCRYPT)*, volume 4515, pages 52–78. 2007.
- [LP08] Yehuda Lindell and Benny Pinkas. Secure multi-party computation for privacy-preserving data mining. In *Crypto ePrint Archive*, 2008.
- [LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao’s protocol for two-party computation. *J. Cryptol.*, 22:161–188, April 2009.

- [LT05] Hsiao-Ying Lin and Wen-Guey Tzeng. An efficient solution to the millionaires' problem based on homomorphic encryption. In *Applied Cryptography and Network Security*, pages 456–466, 2005.
- [MNPS04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, and Yaron Sella. Fairplay-secure two-party computation system. In *USENIX Security Symposium*, pages 287–302, 2004.
- [NP01] Moni Naor and Benny Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms*, pages 448–457, 2001.
- [Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, pages 223–238, 1999.
- [RR07] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *NIPS*, 2007.
- [SG11] Anima Singh and John Guttag. Cardiovascular risk stratification using non-symmetric entropy-based classification trees. In *NIPS workshop on personalized medicine*, 2011.
- [SG13] Anima Singh and John Guttag. Leveraging hierarchical structure in diagnostic codes for predicting incident heart failure. In *ICML workshop on role of machine learning in transforming healthcare*, 2013.
- [SSW09] Ahmad-Reza Sadeghi, Thomas Schneider, and Immo Wehrenberg. Efficient privacy-preserving face recognition. In *Information, Security and Cryptology (ICISC)*, pages 229–244, 2009.
- [SV11] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. Cryptology ePrint Archive, Report 2011/133, 2011.
- [TRMP12] Sebastian Tschiatschek, Peter Reinprecht, Manfred Mücke, and Franz Pernkopf. Bayesian network classifiers with reduced precision parameters. In *Machine Learning and Knowledge Discovery in Databases*, pages 74–89, 2012.
- [Veu11] Thijs Veugen. Comparing encrypted data. <http://msp.ewi.tudelft.nl/sites/default/files/Comparingencrypteddata.pdf>, 2011.
- [VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages I–511, 2001.
- [VKC08] Jaideep Vaidya, Murat Kantarcioglu, and Chris Clifton. Privacy-preserving naive bayes classification. *The International Journal on Very Large Data Bases*, 17(4):879–898, 2008.
- [WGH12] Jenna Wiens, John Guttag, and Eric Horvitz. Learning evolving patient risk processes for c. diff colonization. In *ICML*, 2012.
- [WY04] Rebecca Wright and Zhiqiang Yang. Privacy-preserving bayesian network structure computation on distributed heterogeneous data. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 713–718, 2004.
- [Yao82] Andrew C. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.
- [ZW05] Zhiqiang Yang, Sheng Zhong, and Rebecca N Wright. Privacy-preserving classification of customer data without loss of accuracy. In *SIAM International Conference on Data Mining (SDM)*, 2005.

A Comparison protocols

A.1 Comparison with unencrypted inputs

Our protocol for comparing with encrypted inputs is Protocol 7 and here is some intuition. We follow the main idea from Veugen [Veu11] (found in Section 2.1): compute $2^l + b - a$ (over encrypted data) and check the $l + 1$ -th bit (the bit corresponding to the power 2^l). If it is 1, it means that $b \geq a$, else $b < a$.

We also assume that the encryption scheme is additively homomorphic. In [Veu11] (Section 2.1), Veugen presents a solution for a similar problem except that A only gets the encrypted bit, not in the clear. So we modify his protocol in Protocol 7.

In the description of protocol 7, N is the modulus associated with Paillier's cryptosystem.

Protocol 7 Comparing encrypted data

Input A: $\llbracket a \rrbracket$ and $\llbracket b \rrbracket$, the bit length l of a and b , the secret key SK_{QR} , public key PK_P

Input B: Secret key SK_P , public key PK_{QR} , the bit length l

Output A: $(a \leq b)$

- 1: A: $\llbracket x \rrbracket \leftarrow \llbracket b \rrbracket \cdot \llbracket 2^l \rrbracket \cdot \llbracket a \rrbracket^{-1} \bmod N^2$
 - 2: A chooses a random number $r \leftarrow (0, 2^{\lambda+l}) \cap \mathbb{Z}$
 - 3: A: $\llbracket z \rrbracket \leftarrow \llbracket x \rrbracket \cdot \llbracket r \rrbracket \bmod N^2$ ▷ Blind x
 - 4: A sends $\llbracket z \rrbracket$ to B
 - 5: B decrypts $\llbracket z \rrbracket$
 - 6: A: $c \leftarrow r \bmod 2^l$
 - 7: B: $d \leftarrow z \bmod 2^l$
 - 8: With A, B privately computes the encrypted bit $\llbracket t' \rrbracket$ such that $t = (d < c)$ using DGK
 - 9: A encrypts r_l and sends $\llbracket r_l \rrbracket$ to B
 - 10: B encrypts z_l
 - 11: B: $\llbracket t \rrbracket \leftarrow \llbracket t' \rrbracket \cdot \llbracket z_l \rrbracket \cdot \llbracket r_l \rrbracket$
 - 12: B: sends $\llbracket t \rrbracket$ to A
 - 13: A decrypts and outputs t
-

We will show the correctness of the protocol and then give a proof of security in the honest-but-curious model using modular composition. For the correctness, we just modify the proof of [Veu11].

Proposition A.1. *Protocol 7 is correct and secure in the honest-but-curious model.*

See proof in Appendix C.

A.2 Reversed encrypted comparison

We constructed Protocol 8 which is the same as Protocol 7, except that the roles of A and B are exchanged in Steps 8–13.

Proposition A.2. *Protocol 8 is secure in the honest-but-curious model.*

The proof is in Appendix C.

Protocol 8 Reversed comparing encrypted data

Input A: $\llbracket a \rrbracket$ and $\llbracket b \rrbracket$, public keys PK_{QR} and PK_P

Input B: Secret keys SK_P and SK_{QR}

Output B: $(a \leq b)$

Run Steps 1–7 of Protocol 7.

- 8: With B, A privately computes the encrypted bit $[t']$ such that $t' = (d < c)$ using DGK
 - 9: B encrypts z_l and sends $[z_l]$ to A
 - 10: A encrypts r_l
 - 11: A: $[t] \leftarrow [t'] \cdot [z_l] \cdot [r_l]$
 - 12: A: sends $[t]$ to B
 - 13: B decrypts and outputs t
-

B Preliminaries for proofs

B.1 Secure two-party computation framework

All our protocols are two-party protocols, which we label as party A and party B. In order to show that they do private computations, we work in the honest-but-curious (semi-honest) model as described in [Gol04].

Let $f = (f_A, f_B)$ be a (probabilistic) polynomial function and Π a protocol computing f . A and B want to compute $f(a, b)$ where a is A's input and b is B's input, using Π and with the security parameter λ . The view of party A during the execution of Π is the tuple $V_A(\lambda, a, b) = (1^\lambda; a; r^A; m_1^A, \dots, m_t^A)$ where r is A's random tape and m_1^A, \dots, m_t^A are the messages received by A. We define the view of B similarly. We also define the outputs of parties A and B for the execution of Π on input (a, b) as $\text{Output}_A^\Pi(\lambda, a, b)$ and $\text{Output}_B^\Pi(\lambda, a, b)$, and the global output as $\text{Output}^\Pi(\lambda, a, b) = (\text{Output}_A^\Pi(\lambda, a, b), \text{Output}_B^\Pi(\lambda, a, b))$.

To ensure security, we have to show that whatever A can compute from its interactions with B can be computed from its input and output, which leads us to the following security definition.

Definition B.1. *The two-party protocol Π securely computes the function f if there exists two probabilistic polynomial time algorithms S_A and S_B such that for every possible input a, b of f ,*

$$\begin{aligned} \{S_A(1^\lambda, a, f_A(a, b)), f(a, b)\} &\equiv_c \\ \{V_A(\lambda, a, b), \text{Output}^\Pi(\lambda, a, b)\} \end{aligned}$$

and

$$\begin{aligned} \{S_B(1^\lambda, a, f_B(a, b)), f(a, b)\} &\equiv_c \\ \{V_B(\lambda, a, b), \text{Output}^\Pi(\lambda, a, b)\} \end{aligned}$$

where \equiv_c means computational indistinguishability against probabilistic polynomial time adversaries with negligible advantage in the security parameter λ .

To simplify the notation (and the proofs), hereinafter we omit the security parameter. As we mostly consider deterministic functions f , we can simplify the distributions we want to show being indistinguishable (see [Gol04]): when f is deterministic, to prove the security of Π that computes f , we only have to show that

$$\begin{aligned} S_A(a, f_A(a, b)) &\equiv_c V_A(a, b) \\ S_B(b, f_B(a, b)) &\equiv_c V_B(a, b) \end{aligned}$$

Unless written explicitly, we will always prove security using this simplified definition.

B.2 Modular Sequential Composition

In order to ease the proofs of security, we use sequential modular composition, as defined in [Can98]. The idea is that the parties run a protocol Π and use calls to an ideal functionality f in Π (e.g. A and B compute f privately by sending their inputs to a trusted third party and receiving the result). If we can show that Π respects privacy in the honest-but-curious model and if we have a protocol ρ that privately computes f in the same model, then we can replace the ideal calls for f by the execution of ρ in Π ; the new protocol, denoted Π^ρ is then secure in the honest-but-curious model.

We call *hybrid model with ideal access to f_1, \dots, f_m* or (f_1, \dots, f_m) -*hybrid model* the semi-honest model augmented with an incorruptible trusted party T for evaluating functionalities f_1, \dots, f_m . The parties run a protocol Π that contain calls to T for the evaluation of one of f_1, \dots, f_m . For each call, each party sends its input and wait until the trusted party sends the output back. We emphasize on the fact that the parties must not communicate until receiving T 's output (we consider only *sequential* composition). Ideal calls to the trusted party can be done several times, even for the same function, but each call is independent: T does not maintain state between two calls.

Let Π be a two-party protocol in the (f_1, \dots, f_m) -hybrid model. Let ρ_1, \dots, ρ_m be real protocols (i.e. protocols in the semi-honest model) computing f_1, \dots, f_m and define $\Pi^{\rho_1, \dots, \rho_m}$ as follows. All ideals calls of Π to the trusted party for f_i is replaced by a real execution of ρ_i : if party P_j has to compute f_i with input x_j , P_j halts, starts an execution of ρ_i with the other parties, gets the result β_j when ρ_i concludes, and continues as if β_j was received from T .

Theorem B.2. [Can98] (Theorem 5) restated as in [LP08] (Theorem 3) – *Let f_1, \dots, f_m be two-party probabilistic polynomial time functionalities and ρ_1, \dots, ρ_m protocols that compute respectively f_1, \dots, f_m in the presence of semi-honest adversaries.*

Let g be a two-party probabilistic polynomial time functionality and Π a protocol that securely computes g in the (f_1, \dots, f_m) -hybrid model in the presence of semi-honest adversaries.

Then $\Pi^{\rho_1, \dots, \rho_m}$ securely computes g in the presence of semi-honest adversaries.

B.3 Cryptographic assumptions

Assumption 1. (Quadratic Residuosity Assumption – from [GM82]) *Let $N = p \times q$ be the product of two distinct odd primes p and q . Let \mathbb{QR}_N be the set of quadratic residues modulo N and \mathbb{QNR}_N be the set of quadratic non residues (i.e. $x \in \mathbb{QNR}_N$ if x is not a square modulo N and its Jacobi symbol is 1).*

$\{(N, \mathbb{QR}_N) : |N| = \lambda\}$ and $\{(N, \mathbb{QNR}_N) : |N| = \lambda\}$ are computationally indistinguishable with respect to probabilistic polynomial time algorithms.

Assumption 2. (Decisional Composite Residuosity Assumption – from [Pai99]) *Let $N = p \times q$, $|N| = \lambda$ be the product of two distinct odd primes p and q . A number z is said to be a N -th residue modulo N^2 if there exists a number $y \in \mathbb{Z}_{N^2}$*

$$z = y^N \bmod N^2$$

N -th residues are computationally indistinguishable from non N -th residues with respect to probabilistic polynomial time algorithms.

For further explanations about the last assumption, used for the FHE scheme, we refer the reader to [BGV12].

Assumption 3. (RLWE) *For security parameter λ , let $f(x) = x^d + 1$ where d is a power of 2. Let $q \geq 2$ be an integer. Let $R = \mathbb{Z}[x]/(f(x))$ and let $R_q = R/qR$. Let χ be a distribution over R . The $\text{RLWE}_{d,q,\chi}$ problem is to distinguish between two distributions: In the first distribution, one samples (a_i, b_i) uniformly from R_q^2 . In the second distribution, one first draws $s \leftarrow R_q$ uniformly and then samples $(a_i, b_i) \in R_q^2$ by sampling $a_i \leftarrow R_q$ uniformly, $e_i \leftarrow \chi$, and setting $b_i = a_i \cdot s + e_i$.*

The $\text{RLWE}_{d,q,\chi}$ assumption is that the $\text{RLWE}_{d,q,\chi}$ problem is infeasible.

C Proofs

C.1 Comparison protocols

Proof of Proposition A.1 . Correctness As a and b are l bits integers, $x = 2^l + b - a$ is a $l + 1$ bits integer and its most significant bit (the $l + 1$ -th bit) is 1 iff $a \leq b$. What protocol 7 actually does is computing this bit. The computations are done over encrypted data, using Paillier's encryption scheme. In the rest of the proof, we will do as if the data were not encrypted under Paillier. The correctness will hold as long as we do not experience carry-overs modulo N . In particular, this implies that $l + 1 + \lambda < \log_2 N$. For operations over bits using QR, we don't have this problem as we are operating on \mathbb{F}_2 .

Again, since x is a $l + 1$ bit number, its most significant bit is $x \div 2^l$ where \div denotes the integer division. We have $x = 2^l(x \div 2^l) + (x \bmod 2^l)$ where $0 \leq (x \bmod 2^l) < 2^l$. As $z = x + r$,

$$\begin{aligned} z &= 2^l(z \div 2^l) + (z \bmod 2^l) \\ &= 2^l((x \div 2^l) + (r \div 2^l)) + ((x \bmod 2^l) + (r \bmod 2^l)) \end{aligned}$$

Hence, $z \div 2^l = x \div 2^l + r \div 2^l$ if $(x \bmod 2^l) + (r \bmod 2^l) < 2^l$ and $z \div 2^l = (x \div 2^l) + (r \div 2^l) + 1$ otherwise. More generally, $z \div 2^l = (x \div 2^l) + (r \div 2^l) + t'$ where $t' = 0 \Leftrightarrow (x \bmod 2^l) + (r \bmod 2^l) < 2^l$.

We can also notice that, if $t' = 0$, $z \bmod 2^l = (x \bmod 2^l) + (r \bmod 2^l)$ and $z \bmod 2^l = (x \bmod 2^l) + (r \bmod 2^l) - 2^l$ otherwise. As a consequence,

$$\begin{aligned} t' = 0 &\Leftrightarrow z \bmod 2^l = (x \bmod 2^l) + (r \bmod 2^l) \\ &\Leftrightarrow z \bmod 2^l \geq (r \bmod 2^l) \end{aligned}$$

In the end, as $x \div 2^l$ is either 0 or 1, we can compute everything modulo 2

$$\begin{aligned} x \div 2^l &= (z \div 2^l) - (r \div 2^l) - t' \pmod{2} \\ &= z_l \oplus r_l \oplus t' \end{aligned}$$

Security We suppose that the encrypted bit $[t']$ is ideally computed (using calls to a trusted party in the hybrid model). We show that the protocol is secure in this model and conclude using the sequential modular composition theorem.

A's view is $V_A = ([a], [b], l, \text{SK}_{QR}, \text{PK}_P; r, \text{coins}; [t])$ where SK_{QR} is the secret key for the QR cryptosystem, PK_P is the public key for Paillier's cryptosystem, and coins are the random coins used for the encryptions of 2^l , r and r_l . Given $([a], [b], l, \text{SK}_{QR}, \text{PK}_P, a \leq b)$, we build the simulator S_A :

1. Compute $[\tilde{t}]$ an encryption of the bit $(a \leq b)$ under QR.
2. Pick $\tilde{r} \leftarrow (0, 2^{\lambda+l}) \cap \mathbb{Z}$.
3. Let $\widetilde{\text{coins}}$ be random coins for two Paillier encryptions and one QR encryption.
4. Output $([a], [b], l, \text{SK}_{QR}, \text{PK}_P; \tilde{r}, \widetilde{\text{coins}}; [\tilde{t}])$

The distributions $V_A([a], [b], l, \text{SK}_{QR}, \text{PK}_P, \text{SK}_P, \text{PK}_P)$ and $S_A([a], [b], \text{SK}_{QR}, \text{PK}_P, a \leq b)$ are exactly the same because the randomness is taken from the same distribution in both cases, and the QR cyphertext encrypts the same bit.

B's view is $V_B = (\text{PK}_{QR}, \text{SK}_P, l, [z]; \text{coins}; [t'], [r_l])$ where coins are the random coins used for the encryption of z_l . The simulator $S_B(\text{PK}_{QR}, \text{SK}_P, l)$ runs as follows:

1. Pick $\tilde{z} \leftarrow (0, 2^{\lambda+l}) \cap \mathbb{Z}$.
2. Encrypt \tilde{z} under Paillier: $[\tilde{z}]$.
3. Generate $[\tilde{t}']$ and $[\tilde{r}_l]$, two encryptions of random bits under QR
4. Let $\widetilde{\text{coins}}$ be random coins for one QR encryption.
5. Output $(\text{PK}_{QR}, \text{SK}_P, l, [\tilde{z}]; \widetilde{\text{coins}}; [\tilde{t}'], [\tilde{r}_l])$

The random tapes coins and $\widetilde{\text{coins}}$ are generated in the exact same manner and independently from any other parameter, so

$$\begin{aligned} & (\text{PK}_{QR}, \text{SK}_P, \llbracket \tilde{z} \rrbracket; \widetilde{\text{coins}}; [\tilde{t}'] [\tilde{r}_l]) \\ &= (\text{PK}_{QR}, \text{SK}_P, \llbracket \tilde{z} \rrbracket; \text{coins}; [\tilde{t}'] [\tilde{r}_l]) \end{aligned}$$

Recall that $z = x + r \bmod N$ where x is an l bits integer and r is an $l + \lambda$ bits integer. But as we chose $l + 1 + \lambda < \log_2 N$, we have $z = x + r$. The distribution of \tilde{z} is statistically indistinguishable from the distribution of z (the distributions are distinguishable with an advantage of $2^{-\lambda}$ at most).

We also directly have that $(\text{SK}_P, \llbracket \tilde{z} \rrbracket) \equiv_s (\text{SK}_P, \llbracket z \rrbracket)$ and as a consequence, as the distribution of \tilde{z} and z is independent from \tilde{t}' and \tilde{r}_l ,

$$\begin{aligned} & (\text{PK}_{QR}, \text{SK}_P, \llbracket \tilde{z} \rrbracket; \text{coins}; [\tilde{t}'], [\tilde{r}_l]) \\ & \equiv_s (\text{PK}_{QR}, \text{SK}_P, \llbracket z \rrbracket; \text{coins}; [\tilde{t}'], [\tilde{r}_l]) \end{aligned}$$

By semantic security of QR,

$$\begin{aligned} & (\text{PK}_{QR}, \text{SK}_P, l, \llbracket z \rrbracket; \text{coins}; [\tilde{t}'], [\tilde{r}_l]) \\ & \equiv_c (\text{PK}_{QR}, \text{SK}_P, l, \llbracket z \rrbracket; \text{coins}; [t'], [r_l]) \end{aligned}$$

and

$$\begin{aligned} & S_B(\text{PK}_{QR}, \text{SK}_P, l) \\ & \equiv_c V_B(\llbracket a \rrbracket, \llbracket b \rrbracket, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P) \end{aligned}$$

We conclude the proof of security using modular sequential composition. We replace the ideal calls for computing the encrypted bit $[t']$ by the provable secure DGK protocol and invoke Theorem B.2 to prove security in the semi-honest model. \square

Proof of Proposition A.2. The proof of security is similar to the one of Proposition A.1. Again we first suppose that $[t']$ is ideally computed (hybrid model).

A's view is $V_A = (\llbracket a \rrbracket, \llbracket b \rrbracket, l, \text{PK}_{QR}, \text{PK}_P; r, \text{coins}; [t'], [z_l])$ where PK_{QR} is the public key for the QR cryptosystem, PK_P is the public key for Paillier's cryptosystem and coins is the random tape used for the Paillier encryptions of r and 2^l , and the QR encryption of r_l .

Given $(\llbracket a \rrbracket, \llbracket b \rrbracket, \text{PK}_{QR}, \text{PK}_P)$, we build the simulator S_A :

1. Pick $\tilde{r} \leftarrow (0, 2^{\lambda+l}) \cap \mathbb{Z}$.
2. Generate $[\tilde{t}']$ and $[\tilde{z}_l]$, two encryptions of random bits under QR
3. Let $\widetilde{\text{coins}}$ be random coins for two Paillier encryptions and one QR encryption.
4. Output $(\llbracket a \rrbracket, \llbracket b \rrbracket, l, \text{PK}_{QR}, \text{PK}_P; \tilde{r}, \widetilde{\text{coins}}; [\tilde{z}_l])$

For both cases (A's view and the simulator S_A), r and \tilde{r} are taken from the same uniform distribution over $(0, 2^{\lambda+l}) \cap \mathbb{Z}$, and coins and $\widetilde{\text{coins}}$ are random tapes of the same length, so

$$\begin{aligned} & S_A(\llbracket a \rrbracket, \llbracket b \rrbracket, \text{PK}_{QR}, \text{PK}_P) \\ &= (\llbracket a \rrbracket, \llbracket b \rrbracket, l, \text{PK}_{QR}, \text{PK}_P; r, \text{coins}; [\tilde{z}_l]) \end{aligned}$$

By semantic security of the QR cryptosystem, we conclude with the computational indistinguishability of S_A and V_A distributions:

$$\begin{aligned} & S_A(\llbracket a \rrbracket, \llbracket b \rrbracket, \text{PK}_{QR}, \text{PK}_P) \\ &= (\llbracket a \rrbracket, \llbracket b \rrbracket, l, \text{PK}_{QR}, \text{PK}_P; r, \text{coins}; [\tilde{z}_l]) \\ &\equiv_c (\llbracket a \rrbracket, \llbracket b \rrbracket, l, \text{PK}_{QR}, \text{PK}_P; r, \text{coins}; [z_l]) \\ &= V_A(\llbracket a \rrbracket, \llbracket b \rrbracket, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P) \end{aligned}$$

B's view is $V_B = (\text{SK}_{QR}, \text{SK}_P, \llbracket z \rrbracket, [t]; \text{coins})$ where SK_{QR} is the secret key for the QR cryptosystem, SK_P is the secret key for Paillier's cryptosystem, and coins are the random coins necessary for the QR encryption of z_l . The simulator $S_B(\text{SK}_{QR}, \text{SK}_P, a \leq b)$ runs as follows:

1. Compute $\llbracket \tilde{t} \rrbracket$ an encryption of the bit $(a \leq b)$ under QR.
2. Pick $\tilde{z} \leftarrow (0, 2^{\lambda+l}) \cap \mathbb{Z}$.
3. Encrypt \tilde{z} under Paillier: $\llbracket \tilde{z} \rrbracket$.
4. Let $\widetilde{\text{coins}}$ be random coins for one QR encryption.
5. Output $(\text{SK}_{QR}, \text{SK}_P, l, \llbracket \tilde{z} \rrbracket, [\tilde{t}]; \widetilde{\text{coins}})$

Once again, the distributions of coins and $\widetilde{\text{coins}}$ are identical:

$$\begin{aligned} & (\text{SK}_{QR}, \text{SK}_P, l, \llbracket \tilde{z} \rrbracket, [\tilde{t}]; \widetilde{\text{coins}}) \\ &= (\text{SK}_{QR}, \text{SK}_P, l, \llbracket \tilde{z} \rrbracket, [\tilde{t}]; \text{coins}) \end{aligned}$$

Recall that $z = x + r$ where x is an l bits integer and r is an $l + \lambda$ bits integer. The distribution of \tilde{z} is statistically indistinguishable from the distribution of z . We also directly have that $(\text{SK}_P, \llbracket \tilde{z} \rrbracket) \equiv_s (\text{SK}_P, \llbracket z \rrbracket)$ and as a consequence, as the distribution of \tilde{z} and z is independent from \tilde{t}' ,

$$\begin{aligned} & (\text{SK}_{QR}, \text{SK}_P, l, \llbracket \tilde{z} \rrbracket, [\tilde{t}]; \text{coins}) \\ & \equiv_s (\text{SK}_{QR}, \text{SK}_P, l, \llbracket z \rrbracket, [\tilde{t}]; \text{coins}) \end{aligned}$$

Moreover, by construction, $(\text{SK}_{QR}, [\tilde{t}]) = (\text{SK}_{QR}, [a < b])$ and

$$\begin{aligned} & (\text{SK}_{QR}, \text{SK}_P, l, \llbracket z \rrbracket, [\tilde{t}]; \text{coins}) \\ &= (\text{SK}_{QR}, \text{SK}_P, l, \llbracket z \rrbracket, [a < v]; \text{coins}). \end{aligned}$$

Finally, we have

$$\begin{aligned} & S_B(\text{SK}_{QR}, \text{SK}_P, a \leq b) \\ & \equiv_s V_B(\llbracket a \rrbracket, \llbracket b \rrbracket, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P). \end{aligned}$$

Again, we conclude the proof of security using modular sequential composition. We replace the ideal calls for computing the encrypted bit $[t']$ by the provable secure DGK protocol and invoke Theorem B.2 to prove security in the semi-honest model. \square

C.1.1 Argmax

Proof of Proposition 4.1. Correctness To prove correctness, we have to show that the following invariant holds: at the end of the loop for iteration i , m is the maximum of $\{a_{\pi(j)}\}_{1 \leq j \leq i}$ and $a_{\pi(i_0)} = m$.

If this holds, at the end of the loop iterations $a_{\pi(i_0)}$ is the maximum of $\{a_{\pi(j)}\}_{1 \leq j \leq k} = \{a_j\}_{1 \leq j \leq k}$, hence $i_0 = \text{argmax}_j a_{\pi(j)}$ and $\pi^{-1}(i_0) = \text{argmax}_j a_j$.

At initialization (line 4), the invariant trivially holds as the family $\{a_{\pi(j)}\}_{1 \leq j \leq i}$ contains only one element.

Suppose the property is true for iteration $i - 1$. Let us distinguish two cases:

- If b_i is true (i.e. $m \leq a_{\pi(i)}$), $\max\{a_{\pi(j)}\}_{1 \leq j \leq i-1} \leq a_{\pi(i)}$, as the invariant holds for the previous iteration, and then $\max\{a_{\pi(j)}\}_{1 \leq j \leq i} = a_{\pi(i)}$.

Then i_0 is set to i , $v_i = a'_i$ and $b_i = 1$. As a consequence, m is set by A to

$$v_i + (b_i - 1) \cdot r_i - b_i \cdot s_i = a'_i - s_i = a_{\pi(i)}$$

We have clearly that $a_{\pi(i_0)} = a_{\pi(i)} = m$ and $m = \max\{a_{\pi(j)}\}_{1 \leq j \leq i}$, the invariant holds at the end of the i -th iteration in this case.

- If b_i is false ($m > a_{\pi(i)}$), $\max\{a_{\pi(j)}\}_{1 \leq j \leq i-1} > a_{\pi(i)}$ and $\max\{a_{\pi(j)}\}_{1 \leq j \leq i} = \max\{a_{\pi(j)}\}_{1 \leq j \leq i-1} = m$. Then i_0 is not changed, v_i is set to m'_i and $b_i = 0$. As a consequence,

$$v_i + (b_i - 1) \cdot r_i - b_i \cdot s_i = m'_i - r_i = m$$

m is unchanged. As both m and i_0 stayed the same and $\max\{a_{\pi(j)}\}_{1 \leq j \leq i} = \max\{a_{\pi(j)}\}_{1 \leq j \leq i-1}$, the invariant holds at the end of the i -th iteration in this case.

Security We prove security in the hybrid model where line 5 of the protocol is ideally executed: we ask a trusted party T to compute the function $f(\llbracket x \rrbracket, \llbracket y \rrbracket, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P)$ in the f -hybrid model where

$$\begin{aligned} & f(\llbracket x \rrbracket, \llbracket y \rrbracket, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P) \\ &= (f_A(x, y, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P); \\ & \quad f_B(\llbracket x \rrbracket, \llbracket y \rrbracket, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P)) \end{aligned}$$

and f computes the function of Protocol 8, i.e. f_A returns nothing and f_B returns the bit $x \leq y$.

We will conclude using Theorem B.2.

A's view is

$$\begin{aligned} V_A = & (\{ \llbracket a_i \rrbracket \}_{i=1}^k, l, \text{PK}_{QR}, \text{PK}_P; \\ & \pi, \{ r_i \}_{i=2}^k, \{ s_i \}_{i=2}^k, \text{coins}; \\ & \{ \llbracket v_i \rrbracket \}_{i=2}^k, \{ \llbracket b_i \rrbracket \}_{i=2}^k, \pi(\arg\max_i a_i)) \end{aligned}$$

where coins is the random tape for encryptions. To simulate A's real view, the simulator S_A does the following on input $(\llbracket a_1 \rrbracket, \dots, \llbracket a_k \rrbracket, l, \text{PK}_{QR}, \text{PK}_P, \arg\max_i a_i)$:

1. Picks a random permutation $\tilde{\pi}$ of $\{1, \dots, k\}$
2. Picks $k - 1$ random integers $\tilde{r}_2, \dots, \tilde{r}_k$ in $(0, 2)^{l+\lambda} \cap \mathbb{Z}$
3. Picks $k - 1$ random integers $\tilde{s}_2, \dots, \tilde{s}_k$ in $(0, 2)^{l+\lambda} \cap \mathbb{Z}$
4. Generates $k - 1$ random Paillier encryptions $\llbracket \tilde{v}_2 \rrbracket, \dots, \llbracket \tilde{v}_k \rrbracket$.
5. Generates $k - 1$ random bits \tilde{b}_i
6. Generate a random tape for $2(k - 1)$ Paillier encryptions $\widetilde{\text{coins}}$
7. Outputs

$$\begin{aligned} & (\{ \llbracket a_i \rrbracket \}_{i=1}^k, l, \text{PK}_{QR}, \text{PK}_P; \\ & \tilde{\pi}, \{ \tilde{r}_i \}_{i=2}^k, \{ \tilde{s}_i \}_{i=2}^k, \widetilde{\text{coins}}; \\ & \{ \llbracket \tilde{v}_i \rrbracket \}_{i=2}^k, \{ \llbracket \tilde{b}_i \rrbracket \}_{i=2}^k, \tilde{\pi}(\arg\max_i a_i)) \end{aligned}$$

We define the following hybrids:

- $H_0 = V_A(\llbracket a_1 \rrbracket, \dots, \llbracket a_k \rrbracket, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P)$
- $H_1 = (\{ \llbracket a_i \rrbracket \}_{i=1}^k, l, \text{PK}_{QR}, \text{PK}_P; \pi, \{ r_i \}_{i=2}^k, \{ s_i \}_{i=2}^k, \text{coins}; \{ \llbracket \tilde{v}_i \rrbracket \}_{i=2}^k, \{ \llbracket b_i \rrbracket \}_{i=2}^k, \pi(\arg\max_i a_i))$
- $H_2 = (\{ \llbracket a_i \rrbracket \}_{i=1}^k, l, \text{PK}_{QR}, \text{PK}_P; \pi, \{ \tilde{r}_i \}_{i=2}^k, \{ \tilde{s}_i \}_{i=2}^k, \widetilde{\text{coins}}; \{ \llbracket \tilde{v}_i \rrbracket \}_{i=2}^k, \{ \llbracket \tilde{b}_i \rrbracket \}_{i=2}^k, \pi(\arg\max_i a_i))$
- $H_3 = S_A(\llbracket a_1 \rrbracket, \dots, \llbracket a_k \rrbracket, l, \text{PK}_{QR}, \text{PK}_P, \arg\max_i a_i)$

By semantic security of Paillier's cryptosystem,

$$\begin{aligned}
& (\{[a_i]\}_{i=1}^k, l, \text{PK}_{QR}, \text{PK}_P; \pi, \{r_i\}_{i=2}^k, \{s_i\}_{i=2}^k; \\
& \quad \{[v_i]\}_{i=2}^k, \{[b_i]\}_{i=2}^k, \pi(\arg\max_i a_i)) \\
& \equiv_c (\{[a_i]\}_{i=1}^k, l, \text{PK}_{QR}, \text{PK}_P; \pi, \{r_i\}_{i=2}^k, \{s_i\}_{i=2}^k; \\
& \quad \{[\tilde{v}_i]\}_{i=2}^k, \{[\tilde{b}_i]\}_{i=2}^k, \pi(\arg\max_i a_i))
\end{aligned}$$

and $H_0 \equiv_c H_1$ as $\pi(\arg\max_i a_i) = i_0$

Given that the \tilde{r}_i , \tilde{s}_i and coins are generated according to the same distribution as r_i , s_i (uniform over $(0, 2)^{l+\lambda} \cap \mathbb{Z}$) and coins (random tape for $2(k-1)$ Paillier encryptions), and that they are completely independent from the \tilde{v}_i or π , the hybrids H_1 and H_2 are equal.

Similarly, the distribution of $(\pi, \pi(\arg\max_i a_i))$ and $(\tilde{\pi}, \tilde{\pi}(\arg\max_i a_i))$ are exactly the same. As π and $\tilde{\pi}$ are independent from the other parameters, we also have $H_2 = H_3$. Hence, we showed that

$$\begin{aligned}
& V_A(\{[a_i]\}_{i=1}^k, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P) \\
& \equiv_c S_A(\{[a_i]\}_{i=1}^k, l, \text{PK}_{QR}, \text{PK}_P, \arg\max_i a_i).
\end{aligned}$$

B's view is

$$V_B = (\text{SK}_P, \text{SK}_{QR}, l; \text{coins}; \{b_i\}_{i=2}^k, \{[m'_i]\}_{i=2}^k, \{[a'_i]\}_{i=2}^k)$$

where coins are the random coins for $k-1$ Paillier cyphertext refresh. The simulator $S_B(\text{SK}_P, \text{SK}_{QR}, l)$ runs as follows:

1. Generates a random permutation $\tilde{\pi}$ of $\{1, \dots, k\}$
2. Set $[\tilde{a}_i] = [i]$
3. Run the protocol with the $[\tilde{a}_i]$ as input data, $\tilde{\pi}$ as the permutation, and same parameters otherwise. Let $(\text{SK}_P, \text{SK}_{QR}, l; \widetilde{\text{coins}}; \{b_i\}_{i=2}^k, \{[\tilde{m}'_i]\}_{i=2}^k, \{[\tilde{a}'_i]\}_{i=2}^k)$ be B's view of this run.
4. Outputs

$$(\text{SK}_P, \text{SK}_{QR}, l; \widetilde{\text{coins}}; \{b_i\}_{i=2}^k, \{[\tilde{m}'_i]\}_{i=2}^k, \{[\tilde{a}'_i]\}_{i=2}^k)$$

Let $p : \{a_i\}_{1 \leq i \leq k} \mapsto \{1, \dots, k\}$ be the function that associates a_i to its rank among the a_i (in ascendent order). Let us fix the permutation π for a while and define the following hybrids:

0. $H_0 = V_B(\{[a_i]\}_{i=1}^k, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P)$
1. $H_1 = V_B(\{[p(a_1)]\}_{i=1}^k, l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P)$

We will show that these hybrids are statistically equal for every permutation π .

As $p(\cdot)$ is a map that does not change the order of the a_i , we have that for all i, j , $a_i \leq a_j \Leftrightarrow p(a_i) \leq p(a_j)$. As a consequence, for a given permutation π , the bits b_i do not change if we replace the a_i by $p(a_i)$. Similarly, the way the a'_i and m'_i are generated for H_0 and H_1 is the same: blinding by adding random noise from $(0, 2^{\lambda+l} \cap \mathbb{Z})$. Thus, $H_0 \equiv_s H_1$.

Now, we want to show that $H_1 \equiv_s S_B(\text{SK}_P, \text{SK}_{QR}, l)$ - we do not fix π anymore. Let π_0 be the permutation such that $p(a_i) = \pi_0(i)$. We can then rewrite H_1 as

$$H_1 = V_B([a_1], \dots, [a_k], l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P)$$

As $\tilde{\pi}$ and $\pi \circ \pi_0$ are statistically indistinguishable, we have $H_1 \equiv_s S_B(\text{SK}_P, \text{SK}_{QR}, l)$: recall that S_B 's output is the view of B when the protocol is run with the set $\{a_i = i\}$ as input set and $\tilde{\pi}$ as the permutation. Hence

$$\begin{aligned}
& V_B([a_1], \dots, [a_k], l, \text{SK}_{QR}, \text{PK}_{QR}, \text{SK}_P, \text{PK}_P) \\
& \equiv_s S_B(\text{SK}_P, \text{SK}_{QR}, l)
\end{aligned}$$

We conclude the proof of security using modular sequential composition. We replace the ideal calls for computing the encrypted bits b_i by the provable secure Protocol 8 and invoke Theorem B.2 to prove security in the semi-honest model. \square

C.2 Changing the encryption scheme

Proof of Proposition 4.2. In this protocol the computed function is probabilistic, and we have to show security according to the full definition (cf. section B.1). The function is f :

$$f(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2) = (\llbracket c \rrbracket_2, \emptyset)$$

For the sake of simplicity, we do not take into account the randomness used for the encryptions of r for A and c' for B. As before, the distribution of these coins for one party is completely independent of the other elements to be taken in account in the simulations, so we just do not mention them in security proof.

A's view is $V_A = (\text{PK}_1, \text{PK}_2, \llbracket c \rrbracket_1; r; \llbracket c' \rrbracket_2)$. A's output is $\llbracket c \rrbracket_2$. The simulator $S_A(\text{PK}_1, \text{PK}_2, \llbracket c \rrbracket_1)$ runs as follows:

1. Picks uniformly at random $\tilde{r} \leftarrow M$ and $\tilde{c}' \leftarrow M$.
2. Generates the encryption $\llbracket \tilde{c}' \rrbracket_2$ of \tilde{c}' under E_2 .
3. Outputs $(\text{PK}_1, \text{PK}_2, \llbracket c \rrbracket_1; \tilde{r}; \llbracket \tilde{c}' \rrbracket_2)$.

r and \tilde{r} are taken from the same distribution, independently from any other parameter, so

$$\begin{aligned} & \{(\text{PK}_1, \text{PK}_2, \llbracket c \rrbracket_1; \tilde{r}; \llbracket \tilde{c}' \rrbracket_2); f(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2)\} \\ &= \{(\text{PK}_1, \text{PK}_2, \llbracket c \rrbracket_1; r; \llbracket \tilde{c}' \rrbracket_2); f(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2)\} \end{aligned}$$

(c' depends on r but does not appear in the previous distributions). By semantic security of scheme E_2 we have that

$$\begin{aligned} & \{(\text{PK}_1, \text{PK}_2, \llbracket c \rrbracket_1; r; \llbracket \tilde{c}' \rrbracket_2); f(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2)\} \\ & \equiv_c \{(\text{PK}_1, \text{PK}_2, \llbracket c \rrbracket_1; r; \llbracket c' \rrbracket_2); \llbracket c \rrbracket_2\} \end{aligned}$$

and so

$$\begin{aligned} & \{S_A(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2), f(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2)\} \\ & \equiv_c \{V_A(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2), \text{Output}(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2)\} \end{aligned}$$

B's view is $V_B = (\text{SK}_1, \text{SK}_2; \llbracket c + r \rrbracket_1)$. We build a simulator $S_B(\text{SK}_1, \text{SK}_2)$:

1. Picks a random $\tilde{c} \leftarrow M$.
2. Encrypt \tilde{c} under E_1 .
3. Outputs $(\text{SK}_1, \text{SK}_2, \llbracket \tilde{c} \rrbracket_1)$.

Again, the distribution of \tilde{c} and $c + r$ are identical, so the real distribution $\{(\text{SK}_1, \text{SK}_2; \llbracket c + r \rrbracket_1); \llbracket c \rrbracket_2\}$ and the ideal distribution $\{(\text{SK}_1, \text{SK}_2; \llbracket \tilde{r} \rrbracket_1); f(\llbracket c \rrbracket_1, \text{PK}_1, \text{PK}_2, \text{SK}_1, \text{SK}_2)\}$ are statistically indistinguishable. \square

C.3 Computing dot products

Proof of Proposition 4.3. As B does not receive any message, its view only consists in its input and its random tape used for the encryptions. Hence the simulator S_B simply generate random coins and

$$S_B(y, \text{SK}_P) = (y, \text{SK}_P; \text{coins}) = V_B(x, y, \text{SK}_P, \text{PK}_P).$$

where rand are the random coins.

A's view is $V_A = (x, \text{PK}_P; r^A; \llbracket y_1 \rrbracket, \dots, \llbracket y_n \rrbracket)$. On input $(x, \text{PK}_P, \llbracket v \rrbracket)$, the simulator S_A does the following:

1. Generates n encryptions of 0 using Paillier: c_1, \dots, c_n .
2. Generates the random coins necessary for a Paillier re-randomization and put them in $\widetilde{\text{coins}}$.
3. Outputs $(x, \text{PK}_P; \widetilde{\text{coins}}; c_1, \dots, c_n)$.

coins and $\widetilde{\text{coins}}$ come from the same distribution, independently from other parameters. Thus,

$$\begin{aligned} & \{(x, \text{PK}_P; \widetilde{\text{coins}}; c_1, \dots, c_n); \llbracket \langle x, y \rangle \rrbracket\} \\ &= \{(x, \text{PK}_P; \text{coins}; c_1, \dots, c_n); \llbracket \langle x, y \rangle \rrbracket\} \end{aligned}$$

and by semantic security of Paillier,

$$\begin{aligned} & \{(x, \text{PK}_P; \text{coins}; c_1, \dots, c_n); \llbracket \langle x, y \rangle \rrbracket\} \\ & \equiv_c \{(x, \text{PK}_P; \text{coins}; \llbracket y_1 \rrbracket, \dots, \llbracket y_n \rrbracket); \llbracket v \rrbracket\} \end{aligned}$$

i.e., when f is $f(x, y, \text{SK}_P, \text{PK}_P) = (\llbracket \langle x, y \rangle \rrbracket, \emptyset)$

$$\begin{aligned} & \{S_A(x, \text{PK}_P, \llbracket v \rrbracket); f(x, y, \text{SK}_P, \text{PK}_P)\} \\ & \equiv_c \{V_A(x, y, \text{SK}_P, \text{PK}_P); \text{Output}(x, y, \text{SK}_P, \text{PK}_P)\} \end{aligned}$$

□

C.4 Classifiers

Hyperplane decision

Proof of Proposition 5.1. The client's view is

$$V_C = (\text{PK}_P, \text{PK}_{QR}, x; \{\llbracket v_i \rrbracket\}_{i=1}^k, i_0).$$

The simulator S_C , on input $(\text{PK}_P, \text{SK}_{QR}, x, k^*)$ where $k^* = \underset{i \in [k]}{\text{argmax}} \langle w_i, x \rangle$ does the following:

1. Generate k random Paillier encryptions $\llbracket \tilde{v}_i \rrbracket$
2. Output $(\text{PK}_P, \text{SK}_{QR}, x; \{\llbracket \tilde{v}_i \rrbracket\}_{i=1}^k, k^*)$

As the index i_0 that the client receives is its output, and as Paillier's cryptosystem is semantically secure, the distributions $S_C = (\text{PK}_P, \text{SK}_{QR}, x; \{\llbracket \tilde{v}_i \rrbracket\}_{i=1}^k, k^*)$ and $V_C = (\text{PK}_P, \text{SK}_{QR}, x; \{\llbracket v_i \rrbracket\}_{i=1}^k, i_0)$ are computationally indistinguishable.

As the server views nothing but its inputs (the server does not receive any message in the hybrid model), we use for the trivial simulator that just outputs its inputs for the proof of security.

As Protocols 1 and 3 are secure in the honest-but-curious model, we obtain the security of the hyperplane decision protocol using modular sequential composition (Theorem B.2). □

Bayes classifier

Proof of Proposition 6.1. The client's view is

$$V_C = (\text{PK}_P, \text{SK}_{QR}, x; \llbracket P \rrbracket, \{\llbracket T_{i,j} \rrbracket\}, i_0).$$

The simulator S_C , on input $(\text{PK}_P, \text{SK}_{QR}, x, i_{max})$ where $i_{max} = \underset{j}{\text{argmax}} \mathbb{P}(C = c_j | X = x)$,

- generates tables of random Paillier encryptions $\llbracket \tilde{P} \rrbracket$ and $\{\llbracket T_{i,j} \rrbracket\}$;
- outputs $(\text{PK}_P, \text{SK}_{QR}, x; \llbracket \tilde{P} \rrbracket, \{\llbracket \tilde{T}_{i,j} \rrbracket\}, i_{max})$.

As the integer i_0 that the client receives is its output, and as Paillier's cryptosystem is semantically secure, the distributions $S_C = (\text{PK}_P, \text{SK}_{QR}, x; \llbracket \tilde{P} \rrbracket, \{\llbracket \tilde{T}_{i,j} \rrbracket\}, i_{max})$ and $V_C = (\text{PK}_P, \text{SK}_{QR}, x; \llbracket P \rrbracket, \{\llbracket T_{i,j} \rrbracket\}, i_0)$ are computationally indistinguishable.

Again, as the server views nothing but its inputs (the server does not receive any message in the hybrid model), we use the trivial simulator that outputs its inputs and the random coins for the encryption for the proof of security.

As Protocol 1 is secure in the honest-but-curious model, we obtain the security of the hyperplane decision protocol using modular sequential composition (Theorem B.2). \square

Decision tree

Proof of Proposition 7.1. The proof of security for the server is very easily obtained using modular sequential composition of the comparison protocol and Protocol 2: in the hybrid model, the client receives nothing but the encrypted result.

For the client also the proof is trivial, using modular sequential composition and the semantical security of QR and of the FHE scheme: the encryptions of bits b_i are computational indistinguishable from random bits whether they are encrypted under QR or the FHE scheme. \square

Functional Encryption with Bounded Collusions via Multi-Party Computation^{*}

Sergey Gorbunov[†] Vinod Vaikuntanathan[‡] Hoeteck Wee[§]

September 5, 2012

Abstract

We construct a functional encryption scheme secure against an a-priori bounded polynomial number of collusions for the class of all polynomial-size circuits. Our constructions require only semantically secure public-key encryption schemes and pseudorandom generators computable by small-depth circuits (known to be implied by most concrete intractability assumptions). For certain special cases such as predicate encryption schemes with public index, the construction requires only semantically secure encryption schemes, which is clearly the minimal necessary assumption.

Our constructions rely heavily on techniques from secure multi-party computation and randomized encodings. All our constructions are secure under a strong, adaptive simulation-based definition of functional encryption.

Keywords: Functional Encryption, Multi-Party Computation, Randomized Encodings.

^{*}A preliminary version of this work appeared in the Proceedings of the 32nd Annual International Conference on Cryptology (CRYPTO 2012).

[†]University of Toronto. Email: sgorbunov@cs.toronto.edu. Supported by NSERC Alexander Graham Bell Graduate Scholarship.

[‡]University of Toronto. Email: vinodv@cs.toronto.edu. Supported by an NSERC Discovery Grant and by DARPA under Agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

[§]George Washington University. Email: hoeteck@alum.mit.edu. Supported by NSF CAREER Award CNS-1237429.

Contents

1	Introduction	1
1.1	Our Results	2
1.2	Overview of Our Constructions	3
1.2.1	Functional Encryption for NC1 Circuits	3
1.2.2	A Bootstrapping Theorem and Functional Encryption for P	5
1.3	Definitions of Functional Encryption	6
1.4	A Perspective: Bounded-Use Garbled Circuits	6
2	Preliminaries	7
2.1	Functional Encryption	7
2.2	Shamir’s Secret Sharing	7
2.3	Public Key Encryption.	8
2.4	Decomposable Randomized Encoding	8
3	Security of Functional Encryption against Bounded Collusions	9
4	Background Constructions	11
4.1	Adaptive, Singleton	11
4.2	Adaptive, “Brute Force”	12
4.3	One-Query General Functional Encryption from Randomized Encoding	14
5	A Construction for NC1 circuits	16
5.1	Our Construction	16
5.1.1	Correctness	17
5.2	Setting the Parameters	18
5.3	Proof of Security	18
6	A Bootstrapping Theorem for Functional Encryption	22
6.0.1	Correctness	23
6.1	Proof of Security	24
7	Yet Another Bootstrapping Theorem Using FHE	26
7.0.1	Correctness and Security	27
A	Relations between Definitions of Functional Encryption	30
A.1	A Simulation-based Definition	30
A.2	An Indistinguishability-Based Definition	31
A.3	Relations Between Definitions	32
B	Probabilistic Proofs	35
B.1	Small Pairwise Intersection	35
B.2	Cover-Freeness	36

1 Introduction

Traditional notions of public-key encryption provide *all-or-nothing* access to data: users who possess the secret key can recover the entire message from a ciphertext, whereas those who do not know the secret key learn nothing at all. While such “black-and-white” notions of encryption have served us well for the past thirty years and are indeed being widely used for secure communications and storage, it is time to move beyond. In particular, the advent of cloud computing and the resulting demand for privacy-preserving technologies demands a much more fine-grained access control mechanism for encrypted data.

Boneh, Sahai and Waters [BSW11] recently formalized the notion of functional encryption towards this end, building on and generalizing a number of previous constructs including (anonymous) identity-based encryption (IBE) [Sha84, BF01, Coc01, BW06], fuzzy IBE [SW05], attribute-based encryption (ABE) [GPSW06, LOS⁺10], and predicate encryption [KSW08, LOS⁺10]. Informally, a functional encryption scheme for a circuit family \mathcal{C} associates secret keys SK_C with every circuit C , and ciphertexts CT with every input x . The owner of the secret key SK_C and the ciphertext CT should be able to obtain $C(x)$, but learn nothing else about the input message x itself.¹ Moreover, security should hold against collusions amongst “key holders”, namely, a collusion of users that hold secret keys $\text{SK}_{C_1}, \dots, \text{SK}_{C_q}$ and an encryption of x should learn nothing else about x apart from $C_1(x), \dots, C_q(x)$.

Functional encryption transparently captures as special cases a number of familiar notions of encryption, such as identity-based encryption (IBE), anonymous IBE, fuzzy IBE, attribute-based encryption and so forth. For example, an identity-based encryption scheme can be seen as a functional encryption scheme for the following family of circuits parametrized by the identity:

$$C_{\text{id}'}(\text{id}, \mu) = \begin{cases} (\text{id}, \mu) & \text{if } \text{id} = \text{id}' \\ (\text{id}, \perp) & \text{otherwise} \end{cases}$$

In a similar vein, fuzzy IBE schemes correspond to a circuit that detects proximity between two strings, and attribute based encryption schemes correspond to circuit that can be computed by Boolean formulas. The central and challenging open question in the study of functional encryption is:

Can we build a functional encryption scheme for the class of all poly-size circuits?

To date, constructions of functional encryption are known only for these limited classes of circuits (see [BF01, Coc01, SW05, GPSW06, KSW08, LOS⁺10] and others). More concretely, the state-of-the-art constructions are limited to predicate encryption schemes, where the predicate itself is computable by a “low complexity” class, such as Boolean formula and inner product over fields, both of which are computable in NC1. In particular, a large part of the difficulty in constructing functional encryption schemes lies in the fact that we typically require security against a-priori unbounded collusions, namely, adversaries who obtain secret keys for an unbounded number of circuits C_1, \dots, C_q . This raises the following natural question: can we build functional encryption schemes for arbitrary circuits for some meaningful relaxation of this security requirement?

¹We do not require the circuit C to be secret throughout this work, and in most literature on functional encryption. For the singular exception, see the work of Shi, Shen and Waters [SSW09].

Functional Encryption for Bounded Collusions. In this work, we initiate a systematic study of functional encryption for *bounded* collusions. We consider a relaxed notion of security where the adversary is given secret keys for an *a-priori bounded number of circuits* C_1, \dots, C_q of her choice (which can be made adaptively). This notion, which we call q -bounded security (or security against q collusions), is a natural relaxation of the strong definition above, and could be sufficient in a number of practical use-case scenarios. Our main result in this paper is a construction of q -bounded secure functional encryption schemes for *arbitrary polynomial-size circuit families* under *mild cryptographic assumptions*.

The question of designing IBE schemes with bounded collusions has been considered in a number of works [DKXY02, CHH⁺07, GLW12]. The functional encryption setting presents us with a significantly richer landscape since (1) a secret key SK_C can be used to obtain (partial) information about many messages, as opposed to IBE where a secret key decrypts only ciphertexts for a single identity, and (2) the partial information is a result of a potentially complex computation on the message itself. Our constructions leverage interesting ideas from the study of (information-theoretic) multi-party computation [BGW88, BMR90, DI05] and randomized encodings [Yao86, IK00, AIK06].

We stress that q -bounded security does not restrict the system from issuing an unbounded number of secret keys. We guarantee security against any adversary that gets hold of at most q keys. Specifically, our security definition achieves security against multiple “independent” collusions, as long as each collusion has size at most q . Indeed, it is not clear how to achieve such a security notion for general circuits even in the stateful setting where the system is allowed to maintain a counter while issuing secret keys (analogous to the early notion of stateful signatures). We note that our construction does not require maintaining any state.

1.1 Our Results

The main result of this work is the construction of a q -query functional encryption scheme for the class of all polynomial-size circuits. Our construction is based on the existence of semantically secure public key encryption schemes, and pseudorandom generators (PRG) computable by polynomials of degree $\text{poly}(\kappa)$, where κ is the security parameter. The former is clearly a necessary assumption, and the latter is a relatively mild assumption which, in particular, is implied by most concrete intractability assumptions commonly used in cryptography, such as ones related to factoring, discrete logarithm, or lattice problems.

An important special case of functional encryption that we will be interested in is *predicate encryption with public index* (which is also called attribute-based encryption by some authors). This corresponds to a circuit family \mathcal{C} parametrized by predicates g and defined as:

$$C_g(\text{ind}, \mu) = \begin{cases} (\text{ind}, \mu) & \text{if } g(\text{ind}) = \text{true} \\ (\text{ind}, \perp) & \text{otherwise} \end{cases}$$

Here, ind is the so-called public index, and μ is sometimes referred to as the payload message. For the special case of predicate encryption schemes with *public index*, our construction handles arbitrary polynomial-size circuits while relying *solely* on the existence of semantically secure public-key encryption schemes, which is clearly the minimal necessary assumption. In particular, we do not need the “bounded-degree PRG” assumption for this construction.

In contrast, functional encryption schemes that handle an unbounded number of secret-key queries are known only for very limited classes of circuit families, the most general being inner

product predicates [KSW08, LOS⁺10, OT10]. In particular, constructing an unbounded-query secure functional encryption scheme for general circuit families is considered a major open problem in this area [BSW11]. As for functional encryption schemes with public index (also referred to as “attribute-based encryption” by some authors) that handle an unbounded number of secret-key queries, there are a handful of constructions for polynomial-size formulas [GPSW06, OSW07], which themselves are a sub-class of NC1 circuits.

We will henceforth refer to a functional encryption scheme that supports arbitrary polynomial-size circuits as a *general functional encryption* scheme. Summarizing this discussion, we show:

Theorem 1.1 (Main Theorem, Informal). *Let κ be a security parameter. Assuming the existence of semantically secure encryption schemes as well as PRGs computable by arithmetic circuits of degree- $\text{poly}(\kappa)$, for every $q = q(\kappa)$, there exists a general functional encryption scheme secure against q secret key queries.*

Corollary 1.2 (Informal). *Let κ be a security parameter. Assuming the existence of semantically secure encryption schemes, for every $q = q(\kappa)$, there exists a general predicate encryption scheme with public index secure against q secret key queries.*

We have so far avoided discussing the issue of which security definition to use for functional encryption. Indeed, there are a number of different definitions in the literature, including both *indistinguishability* style and *simulation* style definitions. In a nutshell, we prove our constructions secure under a strong, adaptive simulation-based definition; see Section 1.3 for details.

1.2 Overview of Our Constructions

We proceed with an overview of our construction of a q -bounded general functional encryption scheme.

Starting point. The starting point of our constructions is the fact, observed by Sahai and Seyalioglu [SS10], that *general* functional encryption schemes resilient against a *single* secret-key query can be readily constructed using the beautiful machinery of Yao’s “garbled circuits” [Yao86] (and in fact, more generally, from randomized encodings [IK00, AIK06]).² The construction given in [SS10] only achieves “selective, non-adaptive” security, where the adversary must specify the input message x before it sees the public key, and the single key query C before it sees the challenge ciphertext. We show how to overcome these limitations and achieve “full adaptive security” (for a single key query) by using techniques from non-committing encryption [CFGN96], while still relying only on the existence of semantically secure encryption schemes. All of our constructions henceforth also achieve full adaptive security.

Building on this, our construction proceeds in two steps.

1.2.1 Functional Encryption for NC1 Circuits

In the first step, we show how to construct a q -query functional encryption scheme for NC1 circuits starting from any 1-query scheme.

²We note that [SS10] is completely insecure for collusions of size two: in particular, given two secret keys $\text{SK}_{0\epsilon}$ and $\text{SK}_{1\epsilon}$, an adversary can derive the SK_C for any other C , and moreover, completely recover x .

We denote a “degree” of a circuit C as the degree of the polynomial computing C in the variables of x . A degree of a circuit family denotes the maximum degree of a circuit in the family. Let D denote the degree of NC1 family. The complexity of our construction will be polynomial in both D and q , where q is the number of secret keys the adversary is allowed to see before he gets the challenge ciphertext. This step does not require any additional assumption (beyond semantically secure public key encryption).

The high level approach is as follows: we will run N independent copies of the 1-query scheme. To encrypt, we will encrypt the views of some N -party MPC protocol computing some functionality related to C (aka “MPC in the head” [IKOS07]). As the underlying MPC protocol, we will rely on the BGW semi-honest MPC protocol without degree reduction (c.f. [DI05, Section 2.2]). We will exploit the fact that this protocol is *completely non-interactive* when used to compute *bounded-degree* functions.

We proceed to sketch the construction. Suppose the encryptor holds input $x = (x_1, \dots, x_\ell)$, the decryptor holds circuit C , and the goal is for the decryptor to learn $C(x_1, \dots, x_\ell)$. In addition, we fix t and N to be parameters of the construction.

- The public keys of the system consists of N independent public keys for the 1-query scheme for the same family $C(\cdot)$. The key generation algorithm associates the decryptor with a random subset $\Gamma \subseteq [N]$ of size $Dt + 1$ and generates secret keys for the public keys MPK_i for $i \in \Gamma$. (Note key generation is already a point of departure from previous q -bounded IBE schemes in [DKXY02, CHH⁺07] where the subset Γ is completely determined by C .)
- To encrypt x , the encryptor first chooses ℓ random polynomials μ_1, \dots, μ_ℓ of degree t with constant terms x_1, \dots, x_ℓ respectively. The encryptor computes CT_i to be the encryption of $(\mu_1(i), \dots, \mu_\ell(i))$ under the i 'th public key, and sends $(\text{CT}_1, \dots, \text{CT}_N)$.
- To decrypt, observe that since $C(\cdot)$ has degree at most D ,

$$P(\cdot) := C(\mu_1(\cdot), \dots, \mu_\ell(\cdot))$$

is a univariate polynomial of degree at most Dt and whose constant term is $C(x_1, \dots, x_\ell)$. Now, upon decrypting CT_i for each $i \in \Gamma$, the decryptor recovers $P(i) = C(\mu_1(i), \dots, \mu_\ell(i))$. It can then recover $P(0) = C(x_1, \dots, x_\ell)$ via polynomial interpolation.

The key question now is: what happens when q of the decryptors collude? Let $\Gamma_1, \dots, \Gamma_q \subseteq [N]$ be the (uniformly random) sets chosen for each of the q secret key queries of the adversary. Whenever two of these sets intersect, the adversary obtains two distinct secret keys for the same public key in the underlying one-query FE scheme. More precisely, for every $j \in \Gamma_1 \cap \Gamma_2$, the adversary obtains two secret keys under the public key MPK_j . Since security of MPK_j is only guaranteed under a single adversarial query, we have to contend with the possibility that in this event, the adversary can potentially completely break the security of the public key MPK_j , and learn a share of the encrypted message x .

In particular, to guarantee security, we require that sets $\Gamma_1, \dots, \Gamma_q$ have *small pairwise intersections* which holds for a uniformly random choice of the sets under an appropriate choice of the parameters t and N . With small pairwise intersections, the adversary is guaranteed to learn at most t shares of the input message x , which together reveal no information about x .

For technical reasons, this is not sufficient to establish security of the basic scheme. The first issue, which already arises for a single key query, is that we need to randomize the polynomial P

by adding a random share of 0; this is needed to ensure that the evaluations of P correspond to a random share of $C(x_1, \dots, x_\ell)$, and indeed, the same issue also arises in the BGW protocol. More generally, we need to rerandomize the polynomial P for each of the q queries C_1, \dots, C_q , in order to ensure that it is consistent with random shares of $C_i(x_1, \dots, x_\ell)$, for $i = 1, 2, \dots, q$. This can be done by having the encryptor hard-code additional randomness into the ciphertext. For more details, see Section 5.

Predicate encryption with public index. We point out that this construction also gives us for free a predicate encryption scheme with public index for *arbitrary polynomial-size* circuits (with no a-priori bound on the degree). In this setting, it suffices to realize the following family of circuits parametrized by predicates g :

$$C_g(\text{ind}, \mu) = \begin{cases} (\text{ind}, \mu) & \text{if } g(\text{ind}) = 1 \\ (\text{ind}, 0) & \text{otherwise} \end{cases}$$

We can write C_g as:

$$C_g(\text{ind}, \mu) = (\text{ind}, \mu \cdot g(\text{ind}))$$

Since ind is always part of the output, we can just publish ind “in the clear”. Now, observe that for all ind , C_g , we have $C_g(\text{ind}, \mu)$ is a degree one function in the input μ .

To obtain a *predicate encryption scheme with public index*, we observe that the construction above satisfies a more general class of circuits. In particular, if the input to the encryption algorithm is composed of a *public input* (that we do not wish to hide) and a *secret input* (that we do wish to hide), then the construction above only requires that the circuit C has small degree in the bits of the secret input. Informally, this is true because we do not care about hiding the public input, and thus, we will not secret share it in the construction above. Thus, the degree of the polynomial $P(\cdot)$ grows only with the degree of C in its secret inputs. The bottom line is that since predicate encryption schemes with *public index* deal with circuits that have very low degree in the secret input (degree 1, in particular), our construction handles arbitrary predicates.

1.2.2 A Bootstrapping Theorem and Functional Encryption for P

In the second step, we show a “bootstrapping theorem” for functional encryption schemes. In a nutshell, this shows how to generically convert a q -query secure functional encryption scheme for NC1 circuits into one that is q -query secure for arbitrary polynomial-size circuits, assuming in addition the existence of a pseudo-random generator (PRG) that can be computed with circuits of degree $\text{poly}(\kappa)$. Such PRGs can be constructed based on most concrete intractability assumptions such as those related to factoring, discrete logarithms and lattices.

The main tool that enables our bootstrapping theorem is the notion of randomized encodings [Yao86, IK00, AIK06]. Instead of using the FE scheme to compute the (potentially complicated) circuit C , we use it to compute its randomized encoding \tilde{C} which is typically a much easier circuit to compute. In particular, secret keys are generated for \tilde{C} and the encryption algorithm for the bounded-degree scheme is used to encrypt the pair $(x; R)$, where R is a uniformly random string. The rough intuition for security is that the randomized encoding $\tilde{C}(x; R)$ reveals “no more information than” $C(x)$ itself and thus, this transformation does not adversely affect the security of the scheme.

Unfortunately, intuitions can be misleading and so is this one. Note that in the q -query setting, the adversary obtains not just a single randomized encoding, but q of them, namely $\widetilde{C}_1(x; R), \dots, \widetilde{C}_q(x; R)$. Furthermore, since all these encodings use *the same randomness* R , the regular notion of security of randomized encodings does not apply *as-is*. We solve this issue by hard-coding a large number of random strings (proportional to q) in the ciphertext and using a cover-free set construction, ensuring that the adversary learns q randomized encodings with independently chosen randomness. See Section 6 for more details.

Putting this construction together with a randomized encoding scheme for polynomial-size circuits (which follows from Yao’s garbled circuits [Yao86, AIK06]) whose complexity is essentially the complexity of computing a PRG, we get our final FE scheme.

As a bonus, we show a completely different way to bootstrap q -query FE schemes for NC1 circuits into a q -query FE scheme for any polynomial-size circuits, using a fully homomorphic encryption scheme [Gen09, BV11]. See appendix 7 for more details.

1.3 Definitions of Functional Encryption

Our constructions are shown secure under a strong simulation-based definition, in both the adaptive and non-adaptive sense. The non-adaptive variant requires the adversary to make all its secret key queries before receiving the challenge ciphertext whereas in the adaptive variant, there is no such restriction. Although the adaptive variant is clearly stronger, Boneh, Sahai and Waters [BSW11] recently showed that it is also impossible to achieve, even for very simple circuit families (related to IBE). We observe that the BSW impossibility result holds only if the adversary obtains an unbounded number of ciphertexts (essentially because of a related lower bound for non-committing encryption schemes with unbounded messages). Faced with this state of affairs, we show our constructions are shown secure in the non-adaptive sense, as well as in the adaptive sense with a bounded number of messages.

In addition, we show a number of implications between different variants of these definitions; see Section 3 and Appendix A for more details.

1.4 A Perspective: Bounded-Use Garbled Circuits

The reason why the construction of Sahai and Seyalioglu only achieves security against collusions of size 1 is intimately related to the fact that Yao’s garbled circuits become completely insecure when used more than once. Our constructions may be viewed as a stateless variant of Yao’s garbled circuit that can be reused for some a-priori bounded number of executions. Fix two-parties inputs to be C and x . We can view the ciphertext as encoding of a “universal” circuit of $U_x(\cdot)$ on some fixed input value x , such that we can “delegate” computation on q different inputs C_1, \dots, C_q without leaking any information about x beyond $C_1(x), \dots, C_q(x)$.

Organization of the Paper. We describe the preliminaries and a simulation-based definition of functional encryption in Sections 2 and 3, respectively. For completeness, we describe a construction for 1-query functional encryption and prove its security in the adaptive setting in Section 4. Readers familiar with this construction can go ahead to the next section. We describe our Construction 1 for NC1 circuits in Section 5 and our Construction 2 for bootstrapping in Section 6. An additional FHE-based Construction 3 for bootstrapping is presented in Section 7. An interested reader is referred to the appendices for the definitional implications.

2 Preliminaries

Notations. Let \mathcal{D} denote a distribution over some finite set S . Then, $x \leftarrow \mathcal{D}$ is used to denote the fact that x is chosen from the distribution \mathcal{D} . When we say $x \xleftarrow{\$} S$, we simply mean that x is chosen from the uniform distribution over S . Unless explicitly mentioned, all logarithms are to base 2. For $n \in \mathbb{N}$, let $[n]$ denote the set of numbers $1, \dots, n$. Let κ denote the security parameter.

2.1 Functional Encryption

Let $\mathcal{X} = \{\mathcal{X}_\kappa\}_{\kappa \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$ denote ensembles where each \mathcal{X}_κ and \mathcal{Y}_κ is a finite set. Let $\mathcal{C} = \{\mathcal{C}_\kappa\}_{\kappa \in \mathbb{N}}$ denote an ensemble where each \mathcal{C}_κ is a finite collection of circuits, and each circuit $C \in \mathcal{C}_\kappa$ takes as input a string $x \in \mathcal{X}_\kappa$ and outputs $C(x) \in \mathcal{Y}_\kappa$.

A functional encryption scheme \mathcal{FE} for \mathcal{C} consists of four algorithms $\mathcal{FE} = (\text{FE.Setup}, \text{FE.Keygen}, \text{FE.Enc}, \text{FE.Dec})$ defined as follows.

- **Setup** $\text{FE.Setup}(1^\kappa)$ is a p.p.t. algorithm takes as input the unary representation of the security parameter and outputs the master public and secret keys (MPK, MSK).
- **Key Generation** $\text{FE.Keygen}(\text{MSK}, C)$ is a p.p.t. algorithm that takes as input the master secret key MSK and a circuit $C \in \mathcal{C}_\kappa$ and outputs a corresponding secret key SK_C .
- **Encryption** $\text{FE.Enc}(\text{MPK}, x)$ is a p.p.t. algorithm that takes as input the master public key MPK and an input message $x \in \mathcal{X}_\kappa$ and outputs a ciphertext CT.
- **Decryption** $\text{FE.Dec}(\text{SK}_C, \text{CT})$ is a deterministic algorithm that takes as input the secret key SK_C and a ciphertext CT and outputs $C(x)$.

Definition 2.1 (Correctness). *A functional encryption scheme \mathcal{FE} is correct if for all $C \in \mathcal{C}_\kappa$ and all $x \in \mathcal{X}_\kappa$,*

$$\Pr \left[\begin{array}{l} (\text{MPK}, \text{MSK}) \leftarrow \text{FE.Setup}(1^\kappa); \\ \text{FE.Dec}(\text{FE.Keygen}(\text{MSK}, C), \text{FE.Enc}(\text{MPK}, x)) \neq C(x) \end{array} \right] = \text{negl}(\kappa)$$

where the probability is taken over the coins of FE.Setup , FE.Keygen , and FE.Enc .

Refer to Section 3 for the security definition.

2.2 Shamir's Secret Sharing

We assume familiarity with Shamir's secret-sharing scheme [Sha79] which works as follows: Let \mathbb{F} be a finite field and let $\mathbf{x} = (x_1, \dots, x_n)$ be a vector of any distinct non-zero elements of \mathbb{F} , where $n < |\mathbb{F}|$. Shamir's t -out-of- n secret-sharing scheme works as follows:

- To share a secret $M \in \mathbb{F}$, the sharing algorithm $\text{SS.Share}_{t,n}(M)$ chooses a random univariate polynomial $\mu(x)$ of degree t with constant coefficient M . The n shares are $\mu(x_1), \dots, \mu(x_n)$. Note that any t or fewer shares look uniformly random.
- The reconstruction algorithm SS.Reconstruct takes as input $t + 1$ shares and uses Lagrange interpolation to find a unique degree- t polynomial $\mu(\cdot)$ that passes through the share points. Finally, it computes $\mu(0)$ to recover the secret.

An important property of this scheme is that it permits computation on the shares, a feature used in many multi-party computation protocols starting from [BGW88]. In particular, adding shares gives us $\mu_1(i) + \mu_2(i) = (\mu_1 + \mu_2)(i)$ meaning that that sharing scheme is additively homomorphic. Multiplying shares gives us $\mu_1(i)\mu_2(i) = (\mu_1\mu_2)(i)$ meaning that the scheme is also multiplicatively homomorphic (where $\mu_1\mu_2$ denotes the product of the polynomials). The main catch is that the degree of the polynomial increases with the number of multiplications, requires more shares to recover the answer post multiplication. In other words, the scheme per se is multiplicatively homomorphic for a bounded number of multiplications (but an arbitrary number of additions).

2.3 Public Key Encryption.

A public key encryption scheme $\mathcal{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$, over message space $\mathcal{M} = \{\mathcal{M}_\kappa\}_{\kappa \in \mathbb{N}}$, is a triple of PPT algorithms as follows.

- **Setup.** $\text{PKE.Setup}(1^\kappa)$: takes a unary representation of the security parameter and outputs public and private secret keys (PK, SK) .
- **Encryption.** $\text{PKE.Enc}_{\text{PK}}(M)$: takes the public encryption key PK and a message $M \in \mathcal{M}_\kappa$ and outputs a ciphertext CT .
- **Decryption.** $\text{PKE.Dec}_{\text{SK}}(\text{CT})$: takes the secret key SK and a ciphertext CT and outputs a message $M^* \in \mathcal{M}_\kappa$.

Correctness and security against chosen plaintext attacks are defined as follows.

Definition 2.2. A public key encryption scheme \mathcal{PKE} is correct if for all M ,

$$\Pr[(\text{PK}, \text{SK}) \leftarrow \text{PKE.Setup}(1^\kappa); \text{PKE.Dec}_{\text{SK}}(\text{PKE.Enc}_{\text{PK}}(M)) \neq M] = \text{negl}(\kappa),$$

where the probability is over the coins of PKE.Setup , PKE.Enc .

Definition 2.3. A public key encryption scheme \mathcal{PKE} is (t, ϵ) -IND-CPA secure if for any adversary \mathcal{A} that runs in time t it holds that

$$\left| \Pr[\mathcal{A}^{\text{PKE.Enc}_{\text{PK}}(\cdot)}(1^\kappa, \text{PK}) = 1] - \Pr[\mathcal{A}^{\text{PKE.Enc}_{\text{PK}}(0)}(1^\kappa, \text{PK}) = 1] \right| \leq \epsilon,$$

where the probability is over $(\text{PK}, \text{SK}) \leftarrow \text{PKE.Setup}(1^\kappa)$, the coins of PKE.Enc and the coins of the adversary \mathcal{A} .

2.4 Decomposable Randomized Encoding

Let \mathcal{C} be a circuit that takes inputs $k \in \{0, 1\}^\ell$, $x \in \{0, 1\}^n$ and outputs $\mathcal{C}(k, x) \in \{0, 1\}^m$. A decomposable randomized encoding scheme \mathcal{RE} consists of two algorithms $(\text{RE.Encode}, \text{RE.Decode})$ satisfying the following properties:

1. **Decomposable Encoding.** $\text{RE.Encode}(1^\kappa, \mathcal{C}, x)$: A p.p.t. algorithm takes as inputs a security parameter, a description of a circuit \mathcal{C} , an input x and outputs a randomized encoding:

$$(\tilde{\mathcal{C}}_1(\cdot, x; R), \dots, \tilde{\mathcal{C}}_\ell(\cdot, x; R)) \text{ for } i \in [\ell], \text{ where } \tilde{\mathcal{C}}_i(\cdot, x; R) \text{ depends only on } k_i$$

2. **Decoding.** $\text{RE.Decode}((\tilde{y}_i)_{i=1}^\ell)$: On input of an encoding of a circuit $\tilde{y}_i = \mathcal{C}_i(k_i, x; R)$ for some $k = (k_1, \dots, k_\ell)$ output $\mathcal{C}(k, x)$.
3. **Semantic Security.** We say decomposable randomized encoding \mathcal{RE} is secure if there exists a p.p.t. simulator RE.Sim , such that for every p.p.t. adversary A the outputs of the following two distributions are computationally indistinguishable:

$\text{Exp}_{\mathcal{RE}, A}^{\text{real}}(1^\kappa)$:	$\text{Exp}_{\mathcal{RE}, \text{RE.Sim}}^{\text{ideal}}(1^\kappa)$:
1: $(\mathcal{C}, k = (k_1, \dots, k_\ell), x) \leftarrow A(1^\kappa)$ 2: $(\tilde{\mathcal{C}}_i(\cdot, x; R))_{i=1}^\ell \leftarrow \text{RE.Encode}(1^\kappa, \mathcal{C}, x)$ 3: Output $(\tilde{\mathcal{C}}_i(k_i, x; R))_{i=1}^\ell$	1: $(\mathcal{C}, k = (k_1, \dots, k_\ell), x) \leftarrow A(1^\kappa)$ 2: $(\tilde{\mathcal{C}}_i(k_i, x; R))_{i=1}^\ell \leftarrow \text{RE.Sim}(1^\kappa, \mathcal{C}, \mathcal{C}(k, x))$ 3: Output $(\tilde{\mathcal{C}}_i(k_i, x; R))_{i=1}^\ell$

Note that such a randomized encoding for arbitrary polynomial-size circuits follows from Yao's garbled circuit construction [Yao86, AIK06].

3 Security of Functional Encryption against Bounded Collusions

In this section, we first describe simulation-based definitions for functional encryption with *bounded* collusions, largely based on the recent works of Boneh, Sahai and Waters [BSW11] and O'Neill [O'N10]. We then go on to discuss relations between various flavors of these definitions, with details in Appendix A.

Definition 3.1 (q -NA-SIM- and q -AD-SIM- Security). *Let \mathcal{FE} be a functional encryption scheme for a circuit family $\mathcal{C} = \{\mathcal{C}_\kappa : \mathcal{X}_\kappa \rightarrow \mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$. For every p.p.t. adversary $A = (A_1, A_2)$ and a p.p.t. simulator $S = (S_1, S_2)$, consider the following two experiments:*

$\text{Exp}_{\mathcal{FE}, A}^{\text{real}}(1^\kappa)$:	$\text{Exp}_{\mathcal{FE}, S}^{\text{ideal}}(1^\kappa)$:
1: $(\text{MPK}, \text{MSK}) \leftarrow \text{FE.Setup}(1^\kappa)$ 2: $(x, st) \leftarrow A_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$ 3: $\text{CT} \leftarrow \text{FE.Enc}(\text{MPK}, x)$ 4: $\alpha \leftarrow A_2^{\mathcal{O}(\text{MSK}, \cdot)}(\text{MPK}, \text{CT}, st)$ 5: Output (α, x)	1: $(\text{MPK}, \text{MSK}) \leftarrow \text{FE.Setup}(1^\kappa)$ 2: $(x, st) \leftarrow A_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$ ▶ Let (C_1, \dots, C_q) be A_1 's oracle queries ▶ Let SK_i be the oracle reply to C_i ▶ Let $\mathcal{V} := \{y_i = C_i(x), C_i, \text{SK}_i\}$. 3: $(\text{CT}, st') \leftarrow S_1(\text{MPK}, \mathcal{V}, 1^{ x })$ 4: $\alpha \leftarrow A_2^{\mathcal{O}'(\text{MSK}, st', \cdot)}(\text{MPK}, \text{CT}, st)$ 5: Output (α, x)

We distinguish between two cases of the above experiment:

1. The adaptive case, where:

- the oracle $\mathcal{O}(\text{MSK}, \cdot) = \text{FE.Keygen}(\text{MSK}, \cdot)$ and
- the oracle $\mathcal{O}'(\text{MSK}, st', \cdot)$ is the second stage of the simulator, namely $S_2^{U_x(\cdot)}(\text{MSK}, st', \cdot)$ where $U_x(C) = C(x)$ for any $C \in \mathcal{C}_\kappa$.

The simulator algorithm S_2 is stateful in that after each invocation, it updates the state st' which is carried over to its next invocation. We call a simulator algorithm $S = (S_1, S_2)$ admissible if, on each input C , S_2 makes just a single query to its oracle $U_x(\cdot)$ on C itself.

The functional encryption scheme \mathcal{FE} is then said to be q -query simulation-secure for one message against adaptive adversaries (q -AD-SIM-secure, for short) if there is an admissible p.p.t. simulator $S = (S_1, S_2)$ such that for every p.p.t. adversary $A = (A_1, A_2)$ that makes at most q queries, the following two distributions are computationally indistinguishable:

$$\left\{ \text{Exp}_{\mathcal{FE}, A}^{\text{real}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx} \left\{ \text{Exp}_{\mathcal{FE}, S}^{\text{ideal}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}}$$

2. The non-adaptive case, where the oracles $\mathcal{O}(\text{MSK}, \cdot)$ and $\mathcal{O}'(\text{MSK}, st, \cdot)$ are both the “empty oracles” that return nothing: the functional encryption scheme \mathcal{FE} is then said to be q -query simulation-secure for one message against non-adaptive adversaries (q -NA-SIM-secure, for short) if there is a p.p.t. simulator $S = (S_1, \perp)$ such that for every p.p.t. adversary $A = (A_1, A_2)$ that makes at most q queries, the two distributions above are computationally indistinguishable.

Intuitively, our security definition states that any information that the adversary is able to learn from the ciphertext and secret keys, can be obtained by a simulator from the secret keys and the outputs of the circuit alone. A number of remarks on this definition are in order.

1. In the *non-adaptive* setting, the simulator
 - (a) is *not* allowed to “program” the public parameters or the pre-ciphertext secret key queries;
 - (b) given the real public parameters, adversary’s oracle queries, corresponding real secret keys and circuit output values, is asked to produce a ciphertext indistinguishable from the real ciphertext.
2. In the *adaptive* setting, in addition to the above bullets the second stage simulator
 - (c) is given the real MSK and is allowed to “program” the post-ciphertext secret keys.
3. Even if the the adversary does not request any secret keys, he learns the length of x and therefore, the simulator should be given this information to be on even ground with the adversary. This also ensures that the definition properly generalizes (regular) public-key encryption.
4. We remark that our definitions imply (and are stronger than) those presented in the work of Boneh, Sahai and Waters [BSW11]¹, except we only consider a *single* ciphertext and impose an upper bound on the number of secret key queries.

¹A sketch of the proof is presented in the Appendix A.

Why focus on this definition? First, as mentioned above, our definition is at least as strong as the definition presented in [BSW11]. In addition, in Appendix A we show the following relations between the definitions:

1. *Relations between simulation and indistinguishability:* We show that a *single* message simulation definition implies *single* message indistinguishability definition for both non-adaptive and adaptive worlds.
2. *Relations between single and many messages (simulation):* We show that a *single* message non-adaptive simulation implies *many* messages non-adaptive simulation definition. However, we cannot hope to achieve the same implication for adaptive world due to the impossibility results presented in [BSW11].
3. *Relations between single and many messages (indistinguishability):* Finally, we show that a *single* message indistinguishability implies *many* message indistinguishability definition in both the adaptive and non-adaptive worlds.

These definitional implications are summarized in Figure 1 and proved in Appendix A. As a result of these definitional implications, we focus on proving that our constructions are secure under the *single* message simulation definitions for both adaptive and non-adaptive worlds.

4 Background Constructions

4.1 Adaptive, Singleton

Consider the following simple circuit family that consists of a single identity circuit $\mathcal{C} = \{C\}$, input space $\mathcal{X} = \{0, 1\}$ and $C(x) = x$. We construct a 1-AD-SIM-secure functional encryption for this circuit family, starting from any CPA-secure encryption ($\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec}$). (The construction is inspired by techniques used in non-committing encryption [CFGN96, DN00, KO04].)

- **Setup** $\text{BasicFE.Setup}(1^\kappa)$: Run PKE.Setup twice to generate independent master public-key/secret-key pairs

$$(\text{PK}_i, \text{SK}_i) \leftarrow \text{PKE.Setup}(1^\kappa) \quad \text{for } i = 0, 1$$

Output the master public/secret key pair

$$\text{MPK} := (\text{PK}_0, \text{PK}_1) \quad \text{and} \quad \text{MSK} := (\text{SK}_0, \text{SK}_1)$$

- **Key Generation** $\text{BasicFE.Keygen}(\text{MSK}, C)$: On input the master secret key MSK and a circuit C , pick a random bit $r \xleftarrow{\$} \{0, 1\}$ and output the secret key

$$\text{SK} := (r, \text{SK}_r)$$

- **Encryption** $\text{BasicFE.Enc}(\text{MPK}, x)$: On input the master public key MPK and an input message $x \in \{0, 1\}$: output as ciphertext

$$\text{CT} := (\text{PKE.Enc}(\text{PK}_0, x), \text{PKE.Enc}(\text{PK}_1, x))$$

- **Decryption** $\text{BasicFE.Dec}(\text{SK}, \text{CT})$: On input a secret key $\text{SK} = (r, \text{SK}_r)$ and a ciphertext $\text{CT} = (\text{CT}_0, \text{CT}_1)$, output

$$\text{PKE.Dec}_{\text{SK}_r}(\text{CT}_r)$$

Correctness. Correctness is straight-forward.

Security. We prove that the scheme is 1-AD-SIM-secure. We define a simulator `BasicFE.Sim` that proceeds as follows:

- If the adversary makes a secret key query before seeing the ciphertext, the simulator learns x and can therefore simulate the ciphertext perfectly via normal encryption.
- If the adversary requests for the ciphertext first, then the simulator picks a random bit $\beta \xleftarrow{\$} \{0, 1\}$ and outputs as ciphertext:

$$\text{CT} := (\text{PKE.Enc}(\text{PK}_0, \beta), \text{PKE.Enc}(\text{PK}_1, \bar{\beta}))$$

When the adversary then requests for a secret key, the simulator learns $\text{MSK} = (\text{SK}_0, \text{SK}_1)$ and x , and outputs as the secret key:

$$\text{SK} := (\beta \oplus x, \text{SK}_{\beta \oplus x})$$

We establish security via a series of Games.

Game 0. Normal encryption.

Game 1. If the adversary requests for the ciphertext before making a secret key query, then we modify the ciphertext as follows:

$$\text{CT} := (\text{PKE.Enc}(\text{PK}_0, x \oplus r), \text{PKE.Enc}(\text{PK}_1, x \oplus \bar{r}))$$

Game 2. Output of the simulator.

It is easy to see that the outputs of Games 0 and 1 are computationally indistinguishable by CPA security, and that the outputs of Games 1 and 2 are identically distributed.

Extension to larger \mathcal{X} . It is easy to see that this construction extends to $\mathcal{X} = \{0, 1\}^\lambda$ via λ -wise repetition (that is, λ independent master public keys, etc).

4.2 Adaptive, “Brute Force”

Boneh, et. al [BSW11, Section 4.1] presented a AD-IND-secure scheme for any functionality where the circuit family has polynomial size, starting from any semantically secure public-key encryption scheme. For simplicity, we just write down the construction for a family of two circuits $\mathcal{C} = \{C_0, C_1\}$, which easily extends to any poly-size family. We show that if we replace the underlying encryption scheme with the previous 1-AD-SIM-secure FE encryption for singleton circuit space $\mathcal{C}' = \{C^*\}$, then we obtain a 1-AD-SIM-secure FE encryption for \mathcal{C} .

- **Setup** `BFFE.Setup(1^κ)`: Run `BasicFE.Setup` twice to generate independent master public-key/secret-key pairs

$$(\text{MPK}_i, \text{MSK}_i) \leftarrow \text{BasicFE.Setup}(1^\kappa) \quad \text{for } i = 0, 1$$

Output $(\text{MPK}_0, \text{MPK}_1)$ as the master public key and $(\text{MSK}_0, \text{MSK}_1)$ as the master secret key.

- **Key Generation** $\text{BFFE.Keygen}(\text{MSK}, C_b)$: On input the master secret key MSK and a circuit $C_b \in \mathcal{C}$, output as secret key $\text{SK}_b \leftarrow \text{BasicFE.Keygen}(\text{MSK}_b, C^*)$.
- **Encryption** $\text{BFFE.Enc}(\text{MPK}, x)$: On input the master public key MPK and an input message $x \in \mathcal{X}$, output as ciphertext

$$\text{CT} := (\text{BasicFE.Enc}(\text{MPK}_0, C_0(x)), \text{BasicFE.Enc}(\text{MPK}_1, C_1(x)))$$

- **Decryption** $\text{BFFE.Dec}(\text{SK}_b, \text{CT})$: On input a secret key SK_b and a ciphertext $\text{CT} = (\text{CT}_0, \text{CT}_1)$, output

$$\text{BasicFE.Dec}_{\text{SK}_b}(\text{CT}_b)$$

Correctness. Correctness is straight-forward.

Security. We prove that the scheme is 1-AD-SIM-secure. The simulator BFFE.Sim proceeds as follows:

- If the adversary makes a query C_b before seeing the ciphertext, the simulator learns $C_b(x)$ and then simulates the ciphertext as follows:

$$\text{CT}_b \leftarrow \text{BasicFE.Enc}(\text{MPK}_b, C_b(x)) \quad \text{and} \quad \text{CT}_{1-b} \leftarrow \text{BasicFE.Sim}(\text{MPK}_{1-b}, \emptyset, 1^{|x|})$$

Output $\text{CT} := (\text{CT}_0, \text{CT}_1)$

- If the adversary requests for the ciphertext first, then the simulator simulates the ciphertext as follows:

$$\text{CT}_i \leftarrow \text{BasicFE.Sim}(\text{MPK}_i, \emptyset, 1^{|x|}), \quad \text{for } i = 0, 1$$

Output $\text{CT} := (\text{CT}_0, \text{CT}_1)$. When the adversary then requests for a secret key C_b , the simulator learns $\text{MSK} = (\text{MSK}_0, \text{MSK}_1)$ and $C_b, C_b(x)$ and outputs as secret key

$$\text{SK}_b \leftarrow \text{BasicFE.Sim}(\text{MSK}_b, (C_b(x), C_b), 1^{|x|})$$

We establish security via a series of Games.

Game 0. Normal encryption.

Game 1. Roughly speaking, we will simulate on $\text{MPK}_0, \text{CT}_0$ and follow normal encryption on $\text{MPK}_1, \text{CT}_1$. More precisely, the simulator proceeds as follows:

- If the adversary makes a secret key query C_b before seeing the ciphertext, proceed as follows:
 - if $b = 0$, use the normal encryption for both CT_0 and CT_1 .
 - if $b = 1$, follow BFFE.Sim (that is, generate CT_0 using BasicFE.Sim).
- If the adversary requests for the ciphertext first, then the simulator simulates the ciphertext as follows:

$$\text{CT}_0 \leftarrow \text{BasicFE.Sim}(\text{MPK}_0, \emptyset) \quad \text{and} \quad \text{CT}_1 \leftarrow \text{BasicFE.Enc}(\text{MPK}_1, C_1(x))$$

Output $\text{CT} := (\text{CT}_0, \text{CT}_1)$. When the adversary then requests for a secret key C_b , the simulator proceeds as follows:

- if $b = 0$, follow BFFE.Sim (that is, generate SK_0 using BasicFE.Sim);
- if $b = 1$, follow normal encryption (that is, generate SK_1 using BasicFE.Keygen).

Game 2. Output of the simulator.

It is easy to see that the outputs of Games 0 and 1 are computationally indistinguishable by 1-AD-SIM of the underlying scheme. The same applies to the outputs of Games 1 and 2.

4.3 One-Query General Functional Encryption from Randomized Encoding

Sahai and Seyalioglu [SS10] proved 1-NA-SIM; we observe the same “bootstrapping” construction works for 1-AD-SIM. Let \mathcal{C} be an arbitrary family of poly-size circuits. We construct \mathcal{ONEQFE} scheme for \mathcal{C} as follows.

Let \mathcal{BFFE} denote the brute-force construction defined above. In a high-level the idea is this: suppose we wish to construct an FE scheme for a polynomial-size circuit C and input x . Let $U(C, x)$ denote the universal circuit that output $C(x)$. Let $\tilde{U}(C, x; R)$ denote a randomized encoding of $U(C, x)$ where for every x, R , $\tilde{U}(\cdot, x; R)$ has small locality. Then, assuming C has length λ , we can write

$$\tilde{U}(C, x; R) = (\tilde{U}_1(C[1], x; R), \dots, \tilde{U}_\lambda(C[\lambda], x; R))$$

where $\tilde{U}_i(\cdot, x; R)$ depends only on $C[i]$, the i th bit of circuit C . For each i , we can now use \mathcal{BFFE} scheme for a family of two circuits:

$$\tilde{U}_i := \{\tilde{U}_i(0, \cdot; \cdot), \tilde{U}_i(1, \cdot; \cdot)\}$$

- **Setup** $\text{FE.Setup}(1^\kappa)$: Run the brute-force setup algorithm λ times to generate independent master public-key/secret-key pairs

$$(\text{MPK}_i, \text{MSK}_i) \leftarrow \text{BFFE.Setup}(1^\kappa) \quad \text{for } \tilde{U}_i \text{ and } i = 1, \dots, \lambda$$

Output $(\text{MPK}_i)_{i=1}^\lambda$ as the master public key and $(\text{MSK}_i)_{i=1}^\lambda$ as the master secret key.

- **Key Generation** $\text{FE.Keygen}(\text{MSK}, C)$: On input the master secret key MSK and a circuit $C \in \mathcal{C}$, compute

$$\text{SK}_{C,i} \leftarrow \text{BFFE.Keygen}(\text{MSK}_i, \tilde{U}_i(C[i], \cdot; \cdot)) \quad \text{for } i = 1, \dots, \lambda$$

Output as secret key

$$\text{SK}_C := ((\text{SK}_{C,i})_{i \in [\lambda]})$$

- **Encryption** $\text{FE.Enc}(\text{MPK}, x)$: On input the master public key MPK and an input message $x \in \mathcal{X}$, choose R and compute

$$\text{CT}_i \leftarrow \text{BFFE.Enc}(\text{MPK}_i, (x; R)) \quad \text{for } i = 1, \dots, \lambda$$

Output $(\text{CT}_i)_{i=1}^\lambda$ as the ciphertext.

- **Decryption** $\text{FE.Dec}(\text{SK}_C, \text{CT})$: On input a secret key $\text{SK}_C = (\text{SK}_{C,i})_{i \in [\lambda]}$ and a ciphertext $\text{CT} = (\text{CT}_i)_{i=1}^\lambda$, do the following:

1. Compute $\tilde{y}_i \leftarrow \text{BFFE.Dec}(\text{MSK}_i, \text{CT}_i) = \tilde{U}_i(C[i], x; R)$ for $i = 1, \dots, \lambda$;
2. Run the decoder to get $y \leftarrow \text{RE.Decode}(\tilde{y}_1, \dots, \tilde{y}_\lambda)$.

Output y .

Correctness. Correctness follows directly from the correctness of the brute-force FE construction and randomized encodings.

Security. We first prove that \mathcal{ONEQFE} is 1-NA-SIM-secure (See below on how to modify the proof to show 1-AD-SIM-security). Recall that the simulator gets as input the following values:

1. The public key: $(\text{MPK}_i)_{i=1}^\lambda$;
2. The query C and the corresponding secret key $\text{SK}_C = (\text{SK}_{C,i})_{i=1}^\lambda$;
3. The output of C : $C(x)$;

On the very high level, the security of the scheme follows from the fact that by the security of brute-force construction the adversary can only learn \tilde{y}_i for all i and by the security of the randomized encoding the adversary can only learn $y = C(x)$.

We establish security via a series of Games. Game 0 corresponds to the real experiment and Game $\lambda + 1$ corresponds to the ideal experiment where simulator S produced the ciphertext. The goal of the simulator S is to produce a ciphertext that is indistinguishable from the real ciphertext. Let BFFE.Sim and RE.Sim be the brute-force FE and randomized encoding simulators, respectively.

Game 0. Real encryption experiment.

Game i for $i \in \{1, \dots, \lambda\}$. In Game i , i ciphertexts are encrypted properly using MPK_i and $\lambda - i$ ciphertexts are simulated. Formally, for all $1 \leq j \leq i$, let

$$\text{CT}_i \leftarrow \text{BFFE.Enc}(\text{MPK}_i, (x; R))$$

For all $i < j \leq \lambda$, let

$$\text{CT}_i \leftarrow \text{BFFE.Sim}(\text{MPK}_i, (\tilde{U}_i(C[i], x; R), \tilde{U}_i(C[i], \cdot; \cdot), \text{SK}_{C,i}))$$

Output the ciphertext

$$\text{CT} := (\text{CT}_1, \dots, \text{CT}_\lambda)$$

Game $\lambda + 1$. Same as Game λ , except the randomized encoding is now produced by the RE.Sim. Formally, the simulator S does the following.

1. Let

$$(\tilde{U}_i(C[i], x; R))_{i=1}^\lambda \leftarrow \text{RE.Sim}(1^\kappa, U, U(C, x))$$

2. For all $i \in [\lambda]$, let

$$\text{CT}_i \leftarrow \text{BFFE.Sim}(\text{MPK}_i, (\tilde{U}_i(C[i], x; R), \tilde{U}_i(C[i], \cdot; \cdot), \text{SK}_{C,i}))$$

3. Output the ciphertext

$$\text{CT} := (\text{CT}_1, \dots, \text{CT}_\lambda)$$

Claim 4.0.1. *The outputs of Game 0 and Game λ are computationally indistinguishable.*

Proof. The only different between Games 0 and λ is that in the later the ciphertext produced by the simulator. If there is a distinguisher between the Games, then by we can distinguish between Games i and $i + 1$ for some i , hence compromise the security of the underlying $\mathcal{BF}\mathcal{FE}$ construction. \square

Claim 4.0.2. *The outputs of Game λ and Game $\lambda + 1$ are computationally indistinguishable.*

Proof. This claim follows directly from the security of the randomized encoding simulator. \square

Therefore, we can conclude that the real experiment is indistinguishable from the ideal experiment.

We now sketch how to modify the above proof to show that \mathcal{ONEQFE} is 1-AD-SIM-secure. Construct the simulator $S = (S_1, S_2)$ as follows. The simulator S_1 is the same as in the non-adaptive case, except it passes the simulated decomposable randomized encoding $\tilde{U}(C, x; R)$ as a part of the state to S_2 . Now, assume the oracle query C comes after the challenge ciphertext (the other case is trivial). We invoke the single brute-force simulator BFFE.Sim many times for all MSK_i . For every oracle queries $\tilde{U}_i(C[i], \cdot; \cdot)$ made by BFFE.Sim reply with $\tilde{y}_i \leftarrow \tilde{U}_i(C[i], x; R)$. Finally, output $(\text{SK}_{C,i})_{i \in [\lambda]}$ as the secret key to the adversary.

5 A Construction for NC1 circuits

In this section, we construct a functional encryption scheme for all NC1 circuits secure against q secret-key queries, starting from one that is secure against *a single secret-key query*. Our construction will rely on any semantically secure public-key encryption scheme.

The Class of Circuits. We construct q -bounded FE scheme for a circuit family $\mathcal{C} := \text{NC1}$. In particular, we consider polynomial representation of circuits C in the family. The input message space $\mathcal{X} = \mathbb{F}^\ell$ is an ℓ -tuple of field elements, and for every circuit $C \in \mathcal{C}$, $C(\cdot)$ is an ℓ -variate polynomial over \mathbb{F} of total degree at most D . The complexity of our construction will be polynomial in both D and q , where q is the number of secret keys the adversary is allowed to see before he gets the challenge ciphertext.

5.1 Our Construction

Let $\mathcal{C} := \text{NC1}$ be a circuit family with circuits of degree $D = D(\kappa)$ in its input, and let $q = q(\kappa)$ be a bound on the number of secret key queries. Our scheme is associated with additional parameters $S = S(\kappa)$, $N = N(\kappa)$, $t = t(\kappa)$ and $v = v(\kappa)$ (for an instantiation of the parameters, see Section 5.2).

We start by defining a new family \mathcal{G} as follows:

$$G_{C,\Delta}(x, Z_1, \dots, Z_S) := C(x) + \sum_{i \in \Delta} Z_i \quad (1)$$

where $\Delta \subseteq [S]$ and $Z_1, \dots, Z_S \in \mathbb{F}$.

Let $(\text{OneQFE.Setup}, \text{OneQFE.Keygen}, \text{OneQFE.Enc}, \text{OneQFE.Dec})$ be a functional encryption scheme for \mathcal{G} secure against a *single* secret key query. Our q -query secure encryption scheme $\mathcal{BD}\mathcal{FE} = (\text{BdFE.Setup}, \text{BdFE.Keygen}, \text{BdFE.Enc}, \text{BdFE.Dec})$ for \mathcal{C} works as follows:

- **Setup** $\text{BdFE.Setup}(1^\kappa)$: Run the one-query setup algorithm N times to generate independent master public-key/secret-key pairs

$$(\text{MPK}_i, \text{MSK}_i) \leftarrow \text{OneQFE.Setup}(1^\kappa) \quad \text{for } i = 1, \dots, N$$

Output $(\text{MPK}_i)_{i=1}^N$ as the master public key and $(\text{MSK}_i)_{i=1}^N$ as the master secret key.

- **Key Generation** $\text{BdFE.Keygen}(\text{MSK}, C)$: On input the master secret key MSK and a circuit $C \in \mathcal{C}$,

1. Choose a uniformly random set $\Gamma \subseteq [N]$ of size $tD + 1$;
2. Choose a uniformly random set $\Delta \subseteq [S]$ of size v ;
3. Generate the secret keys

$$\text{SK}_{C,\Delta,i} \leftarrow \text{OneQFE.Keygen}(\text{MSK}_i, G_{C,\Delta}) \quad \text{for every } i \in \Gamma$$

Output as secret key $\text{SK}_C := (\Gamma, \Delta, (\text{SK}_{C,\Delta,i})_{i \in \Gamma})$.

- **Encryption** $\text{BdFE.Enc}(\text{MPK}, x)$: On input the master public key $\text{MPK} = (\text{MPK}_i)_{i=1}^N$ and an input message $x = (x_1, \dots, x_\ell) \in \mathcal{X}$:

1. For $i = 1, 2, \dots, \ell$, pick a random degree t polynomial $\mu_i(\cdot)$ whose constant term is x_i .
2. For $i = 1, 2, \dots, S$, pick a random degree Dt polynomial $\zeta_i(\cdot)$ whose constant term is 0.
3. Run the one-query encryption algorithm OneQFE.Enc N times to produce ciphertexts

$$\text{CT}_i \leftarrow \text{OneQFE.Enc}(\text{MPK}_i, (\mu_1(i), \dots, \mu_\ell(i), \zeta_1(i), \dots, \zeta_S(i))) \quad \text{for } i = 1, \dots, N$$

Output $(\text{CT}_i)_{i=1}^N$ as the ciphertext.

- **Decryption** $\text{BdFE.Dec}(\text{SK}_C, \text{CT})$: On input a secret key $\text{SK}_C = (\Gamma, \Delta, (\text{SK}_{C,\Delta,i})_{i \in \Gamma})$ and a ciphertext $\text{CT} = (\text{CT}_i)_{i=1}^N$, do the following:

1. Compute a degree Dt polynomial $\eta(\cdot)$ such that $\eta(i) = \text{OneQFE.Dec}(\text{SK}_{C,\Delta,i}, \text{CT}_i)$ for all $i \in \Gamma$.
2. Output $\eta(0)$.

5.1.1 Correctness

We show that the scheme above is correct. By correctness of the underlying single-query FE, we have that for all $i \in \Gamma$,

$$\begin{aligned} \eta(i) &= G_{C,\Delta}(\mu_1(i), \dots, \mu_\ell(i), \zeta_1(i), \dots, \zeta_S(i)) \\ &= C(\mu_1(i), \dots, \mu_\ell(i)) + \sum_{a \in \Delta} \zeta_a(i) \end{aligned}$$

Since $|\Gamma| \geq Dt + 1$, this means that η is equal to the degree Dt polynomial

$$\eta(\cdot) = C(\mu_1(\cdot), \dots, \mu_\ell(\cdot)) + \sum_{a \in \Delta} \zeta_a(\cdot)$$

Hence, $\eta(0) = C(x_1, \dots, x_\ell) = C(x)$.

5.2 Setting the Parameters

We show how to set the parameters $S = S(\kappa)$, $N = N(\kappa)$ and $t = t(\kappa)$. These parameters govern the choice of the sets Γ and Δ during the key generation algorithm, and are required to satisfy the following two conditions:

Small Pairwise Intersections. Let $\Gamma_1, \dots, \Gamma_q \subseteq [N]$ be the (uniformly random) sets chosen for each of the q secret key queries of the adversary. Whenever two of these sets intersect, the adversary obtains two distinct secret keys for the underlying one-query secure FE scheme. More precisely, for every $j \in \Gamma_1 \cap \Gamma_2$, the adversary obtains two secret keys under the public key MPK_j . Since security of MPK_j is only guaranteed under a single adversarial query, we have to contend with the possibility that in this event, the adversary can potentially completely break the security of the public key MPK_j . In particular, for every such j , the adversary potentially learns a share of the encrypted input message x .

Thus, to guarantee security, we require that the union of the pairwise intersections of $\Gamma_1, \dots, \Gamma_q$ is small. In particular, we require that $\left| \bigcup_{i \neq j} (\Gamma_i \cap \Gamma_j) \right| \leq t$. This ensures that the adversary learns at most t shares of the input message x , which together reveal no information about x .

A simple probabilistic argument shows that this is true (with probability $1 - 2^{-\Omega(t/q^2)}$) as long as $q^2 \cdot (Dt/N)^2 \cdot N \leq t/10$. In other words, we will set $t(\kappa) = \Theta(q^2 \kappa)$ and $N(\kappa) = \Theta(D^2 q^2 t)$ which satisfies the above constraint with probability $1 - 2^{-\Omega(\kappa)}$. For details, we refer an interested reader to Appendix B.1.

Cover-Freeness. Let $\Delta_1, \dots, \Delta_q \subseteq [S]$ be the (uniformly random) sets of size v chosen for each of the q secret key queries of the adversary. The security proof relies on the condition that the polynomials $\sum_{a \in \Delta_j} \zeta_a(\cdot)$ are uniformly random and independent which is true if the collection of sets $\Delta_1, \dots, \Delta_q$ is cover-free. That is, for every $i \in [q]$: $\Delta_i \setminus \left(\bigcup_{j \neq i} \Delta_j \right) \neq \emptyset$.

A simple probabilistic argument shows that this is true (with probability $1 - 2^{-\Omega(q^2 v^2/S)}$) as long as $q^2 v^2/S \leq v/100$. In other words, we will set $v(\kappa) = \Theta(\kappa)$ and $S(\kappa) = \Theta(vq^2)$ which satisfies the above constraint with probability $1 - 2^{-\Omega(\kappa)}$. For details, we refer an interested reader to Appendix B.2.

We remark that in our construction, multiple secret key queries for the same $C \in \mathcal{C}$ result in different secret keys SK_C , essentially because of the different random choices of the sets Δ and Γ . Using a pseudorandom function (applied to C), it is possible to ensure that multiple secret key queries for the same C result in the same answer.

5.3 Proof of Security

Theorem 5.1. *Let $\text{ON}\mathcal{EQFE}$ be a 1-AD-SIM-secure (resp. 1-NA-SIM-secure) functional encryption scheme for any family of poly-size circuits. Then, for any circuit family \mathcal{C} computable in $NC1$ the BDFE scheme described above is q -AD-SIM-secure (resp. q -NA-SIM-secure).*

We prove that the construction BDFE given in Section 5 is q -AD-SIM-secure if we start out with a 1-AD-SIM-secure scheme. This subsumes the non-adaptive variant of the proof. By Theorem A.1, this implies that BDFE is q -NA-SIM-secure for many messages. However, it is only single-message

q -AD-SIM-secure (see Figure 1 for relations).

We establish security by first defining the simulator and then arguing that its output is indistinguishable via a series of Games. For readability, we adopt the following convention: we use i to index over values in $[N]$, and we use j to index over the queries.

Overview. Suppose the adversary receives the challenge ciphertext after seeing $q^* \leq q$ queries. The simulator has to simulate the ciphertext and answer the remaining secret key queries. We may assume it already knows all of $\Gamma_1, \dots, \Gamma_q, \Delta_1, \dots, \Delta_q$. This is because:

- for $j \leq q^*$, the simulator gets Γ_j, Δ_j from SK_j ;
- for $j > q^*$, the simulator gets to program Γ_j, Δ_j and could pick all these quantities in advance.

We first describe our strategy for simulating the ciphertext $\text{CT} = (\text{CT}_1, \dots, \text{CT}_N)$ and the secret keys. Let \mathcal{I} denote

$$\bigcup_{j \neq j'} (\Gamma_j \cap \Gamma_{j'})$$

We will consider two cases:

- $i \in \mathcal{I}$: Here, we may issue more than one secret key corresponding to $(\text{MPK}_i, \text{MSK}_i)$; therefore, we can no longer rely on the security of the underlying one-query FE scheme. Instead, we rely on the statistical security of the underlying MPC protocol and the fact that $|\mathcal{I}| \leq t$. Specifically, we can simulate CT_i and the secret keys honestly.
- $i \notin \mathcal{I}$: Here, we issue at most one secret key corresponding to $(\text{MPK}_i, \text{MSK}_i)$; this is because at most one of the sets $\Gamma_1, \dots, \Gamma_q$ contains i . Suppose $i \in \Gamma_j$. We may now appeal to the security of the underlying one-query FE scheme. Specifically, we simulate CT_i computationally using the simulator for the underlying one-query FE scheme. If $j \leq q^*$, then we do not need to program secret keys at all. If $j > q^*$, upon receiving query C_j , we program the corresponding keys $\text{SK}_{C_j, \Delta_j, i}$ using the one-query simulator.

We formally define the simulator BdFE.Sim as follows:

Simulating the ciphertext after query q^* . Here, the simulator knows $\Gamma_1, \dots, \Gamma_q, \Delta_1, \dots, \Delta_q$; the queries C_1, \dots, C_{q^*} , the outputs $C_1(x), \dots, C_{q^*}(x)$, and the secret keys $\text{SK}_1, \dots, \text{SK}_{q^*}$.

1. Uniformly and independently sample ℓ random degree t polynomials μ_1, \dots, μ_ℓ whose constant terms are all 0.
2. We sample the polynomials ζ_1, \dots, ζ_S as follows: let $\Delta_0 := \emptyset$. For $j = 1, 2, \dots, q$:
 - (a) by the cover-free property, fix some $a^* \in \Delta_j \setminus (\Delta_0 \cup \dots \cup \Delta_{j-1})$;
 - (b) for all $a \in (\Delta_j \setminus (\Delta_0 \cup \dots \cup \Delta_{j-1})) \setminus \{a^*\}$, set ζ_a to be a uniformly random degree Dt polynomial whose constant term is 0;
 - (c) if $j \leq q^*$, pick a random degree Dt polynomial $\eta_j(\cdot)$ whose constant term is $C_j(x)$; if $j > q^*$, pick random values for $\eta_j(i)$ for all $i \in \mathcal{I}$;

(d) the evaluation of ζ_{a^*} on the points in \mathcal{I} is defined by the relation:

$$\eta_j(\cdot) = C_j(\mu_1(\cdot), \dots, \mu_\ell(\cdot)) + \sum_{a \in \Delta_j} \zeta_a(\cdot)$$

Finally, for all $a \notin (\Delta_1 \cup \dots \cup \Delta_q)$, set ζ_a to be a uniformly random degree Dt polynomial whose constant term is 0.

3. For each $i \in \mathcal{I}$, run the one-query encryption algorithm **OneQFE.Enc** to produce ciphertexts

$$\text{CT}_i \leftarrow \text{OneQFE.Enc}(\text{MPK}_i, (\mu_1(i), \dots, \mu_\ell(i), \zeta_1(i), \dots, \zeta_S(i)))$$

4. For each $i \notin \mathcal{I}$, run the one-query simulator **OneQFE.Sim** to produce ciphertexts CT_i as follows: at most one of $\Gamma_1, \dots, \Gamma_q$ contains i .

- If such a set exists, let j denote the unique set Γ_j that contains i (i.e. $i \in \Gamma_j$). If $j \leq q^*$, compute

$$\text{CT}_i \leftarrow \text{OneQFE.Sim}(\text{MPK}_i, (\eta_j(i), G_{C_j, \Delta_j}, \text{SK}_{C_j, \Delta_j, i}))$$

where $\text{SK}_{C_j, \Delta_j, i}$ is provided as part of SK_j .

- If no such set exist or $j > q^*$, then compute

$$\text{CT}_i \leftarrow \text{OneQFE.Sim}(\text{MPK}_i, \emptyset)$$

Output $(\text{CT}_i)_{i=1}^N$ as the ciphertext.

Simulating secret key SK_j , for $j > q^*$. Here, the simulator gets $\text{MSK} = (\text{MSK}_1, \dots, \text{MSK}_N)$ and $C_j(x)$, C_j and needs to simulate $(\text{SK}_{C_j, \Delta_j, i})_{i \in \Gamma_j}$.

1. For each $i \in \Gamma_j \cap \mathcal{I}$, pick $\text{SK}_{C_j, \Delta_j, i} \leftarrow \text{OneQFE.Keygen}(\text{MSK}_i, G_{C_j, \Delta_j})$.
2. For each $i \in \Gamma_j \setminus \mathcal{I}$ (i.e, Γ_j is the only set that contains i),
 - (a) pick a random degree Dt polynomial $\eta_j(\cdot)$ whose constant term is $C_j(x)$ and subject to the constraints on the values in \mathcal{I} chosen earlier;
 - (b) run $\text{OneQFE.Sim}(\text{MSK}_i, (\eta_j(i), G_{C_j, \Delta_j}))$ to obtain $\text{SK}_{C_j, \Delta_j, i}$ so that CT_i decrypts to $\eta_j(i)$.

Output $(\text{SK}_{C_j, \Delta_j, i})_{i \in \Gamma_j}$.

We establish security via a series of Games. The simulator is described above.

Game 1. We modify $\zeta_1, \dots, \zeta_S, \eta_1, \dots, \eta_q$ to be the same as that in the simulator.

Game 2. We simulate $(\text{CT}_i)_{i \notin \mathcal{I}}$ and $\text{SK}_j, j > q^*$ as in the simulator.

Game 3. The output of the simulator. That is, we modify how polynomials μ_1, \dots, μ_ℓ are sampled.

Claim 5.1.1. *The outputs of Game 0 and Game 1 are identically distributed.*

Proof. In the normal encryption, ζ_{a^*} is chosen at random and $\eta_j(\cdot)$ is defined by the relation. From Step 2 in the ciphertext simulation and Step 2 in the secret keys simulation (for $j > q^*$) BdFE.Sim , essentially, chooses $\eta_j(\cdot)$ at random which defines ζ_{a^*} . It is easy to see that reversing the order of how the polynomials are chosen produces the same distribution. \square

Claim 5.1.2. *The outputs of Game 1 and Game 2 are computationally indistinguishable.*

Proof. Informally, this follows from the security of the underlying one-query FE scheme and the fact that for all $i \notin \mathcal{I}$, we run $\text{OneQFE.Keygen}(\text{MSK}_i, \cdot)$ at most once.

By a hybrid argument, it suffices to show that for all $i \notin \mathcal{I}$, the distribution of CT_i in Game 1 and 2 are computationally indistinguishable (given MPK_i and $\text{SK}_1, \dots, \text{SK}_q$). Indeed, fix such a $i \notin \mathcal{I}$ and a corresponding unique j such that $i \in \Gamma_j$ (the case no such j exists is similar).

First, observe that amongst $\text{SK}_1, \dots, \text{SK}_q$, only SK_j contains a key $\text{SK}_{C_j, \Delta_j, i}$ that is generated using either $\text{SK}_{C_j, \Delta_j, i} \leftarrow \text{OneQFE.Keygen}(\text{MSK}_i, G_{C_j, \Delta_j})$ (for the non-adaptive queries) or $\text{SK}_{C_j, \Delta_j, i} \leftarrow \text{OneQFE.Sim}(\text{MSK}_i, (\eta_j(i), G_{C_j, \Delta_j}))$ (for the adaptive queries).

Case 1: Assume $j \leq q^*$. Observe that

$$\begin{aligned} \eta_j(i) &= C_j(\mu_1(i), \dots, \mu_\ell(i)) + \sum_{a \in \Delta_j} \zeta_a(i) \\ &= G_{C_j, \Delta_j}(\mu_1(i), \dots, \mu_\ell(i), \zeta_1(i), \dots, \zeta_S(i)) \end{aligned} \quad (2)$$

which means that in both Games 1 and 2, CT_i decrypts to the same value. Now, note that in Game 1, CT_i is generated using

$$\text{CT}_i \leftarrow \text{OneQFE.Enc}(\text{MPK}_i, (\mu_1(i), \dots, \mu_\ell(i), \zeta_1(i), \dots, \zeta_S(i)))$$

By the security of the underlying FE scheme, this is computationally indistinguishable from

$$\text{OneQFE.Sim}(\text{MPK}_i, (G_{C_j, \Delta_j}(\mu_1(i), \dots, \mu_\ell(i), \zeta_1(i), \dots, \zeta_S(i)), G_{C_j, \Delta_j}, \text{SK}_{C_j, \Delta_j, i}))$$

By the Equation 2, this is the same as

$$\text{OneQFE.Sim}(\text{MPK}_i, (\eta_j(i), G_{C_j, \Delta_j}, \text{SK}_{C_j, \Delta_j, i}))$$

which is the distribution of CT_i in Game 2.

Case 2: Assume $j > q^*$. Then:

- $\text{CT}_i \leftarrow \text{OneQFE.Sim}(\text{MPK}_i, \emptyset)$ and
- $\text{SK}_{C_j, \Delta_j, i} \leftarrow \text{OneQFE.Sim}(\text{MSK}_i, (\eta_j(i), G_{C_j, \Delta_j}))$

Similarly, by the Equation 2 and by the security of the underlying one-query FE scheme this simulated pair of ciphertext and secret key is indistinguishable from the real. \square

Claim 5.1.3. *The outputs of Game 2 and Game 3 are identically distributed.*

Proof. In Game 2, the polynomials μ_1, \dots, μ_ℓ are chosen with constant terms x_1, \dots, x_ℓ , respectively. In Game 3, these polynomials are now chosen with 0 constant terms. This only affects the distribution of μ_1, \dots, μ_ℓ themselves and polynomials ζ_1, \dots, ζ_S . Moreover, only the evaluations of these polynomials on the points in \mathcal{I} affect the outputs of the games. Now observe that:

- The distribution of the values $\{\mu_1(i), \dots, \mu_\ell(i)\}_{i \in \mathcal{I}}$ are identical in both Game 2 and 3. This is because in both games, we choose these polynomials to be random degree t polynomials (with different constraints in the constant term), so their evaluation on the points in \mathcal{I} are identically distributed, since $|\mathcal{I}| \leq t$.
- The values $\{\zeta_1(i), \dots, \zeta_S(i)\}_{i \in \mathcal{I}}$ depend only on the values $\{\mu_1(i), \dots, \mu_\ell(i)\}_{i \in \mathcal{I}}$.

The claim follows readily from combining these observations. \square

6 A Bootstrapping Theorem for Functional Encryption

In this section, we show a “bootstrapping-type” theorem for functional encryption (FE). In a nutshell, this shows how to take a q -query functional encryption scheme for “bounded degree” circuits, and transform them into a q -query functional encryption scheme for arbitrary polynomial-size circuits. The transformation relies on the existence of a pseudorandom generator (PRG) that stretches the seed by a constant factor, and which can be computed by circuits of degree $\text{poly}(\kappa)$. This is a relatively mild assumption, and in particular, is implied by most concrete intractability assumptions commonly used in cryptography, such as ones related to factoring, discrete logarithm, or lattice problems.

In a high-level the idea is this: Suppose we wish to construct an FE scheme for a family \mathcal{C} of polynomial-size circuit. Let $C \in \mathcal{C}$ and x be some input. Then, let $\tilde{C}(x; R)$ denote a randomized encoding of C that is computable by a constant-depth circuit with respect to the inputs x and R . By [AIK06, Theorem 4.14], we know that assuming the existence of a pseudo-random generator in $\oplus\text{L}/\text{poly}$, such a randomized encoding exists for every polynomial-size circuit C .

Consider a new family of circuits \mathcal{G} defined as follows:

$$G_{C,\Delta}(x, R_1, \dots, R_S) := \tilde{C}\left(x; \bigoplus_{a \in \Delta} R_a\right)$$

Observe the following:

- Since for any C , $\tilde{C}(\cdot; \cdot)$ is computable by a constant-depth circuit, then $G_{C,\Delta}(\cdot; \cdot)$ is computable by a constant-degree polynomial. Using the result from the previous scheme, we have a q -AD-SIM-secure FE scheme for G .
- Given a functional encryption scheme for \mathcal{G} , it is easy to construct one for \mathcal{C} . Decryption works by first recovering the output of $G_{C,\Delta}$ and then applying the decoder for the randomized encoding.

- Informally, 1-AD-SIM-security follows from the fact that the ciphertext together with the secret key reveals only the output of $\tilde{C}(x)$, which in turn reveals no more information than $C(x)$. More formally, given $C(x)$, we can simulate $\tilde{C}(x)$ and then the ciphertext, using first the simulator for the randomized encoding and then that for the underlying FE scheme.
- The role of the subset Δ is similar to that in the preceding construction — to “rerandomize” the randomness used in G , which is necessary to achieve q -AD-SIM-security.

Functional Encryption Scheme for \mathcal{C} . Let $(\text{BdFE.Setup}, \text{BdFE.Keygen}, \text{BdFE.Enc}, \text{BdFE.Dec})$ be a q -AD-SIM-secure scheme for \mathcal{G} , with a simulator BdFE.Sim . We construct an encryption scheme $(\text{FE.Setup}, \text{FE.Keygen}, \text{FE.Enc}, \text{FE.Dec})$ for \mathcal{C} works as follows (that takes parameters S, v as before).

- **Setup** $\text{FE.Setup}(1^\kappa)$: Run the bounded FE setup algorithm to generate a master public-key/secret-key pair $(\text{MPK}, \text{MSK}) \leftarrow \text{BdFE.Setup}(1^\kappa)$.
- **Key Generation** $\text{FE.Keygen}(\text{MSK}, C)$: On input the master secret key MSK and a circuit $C \in \mathcal{C}$, do the following:
 1. Choose a uniformly random set $\Delta \subseteq [S]$ of size v ;
 2. Generate the secret key $\text{SK}_{C,\Delta} \leftarrow \text{BdFE.Keygen}(\text{MSK}, G_{C,\Delta})$.

Output as secret key $\text{SK}_C := (\Delta, \text{SK}_{C,\Delta})$.

- **Encryption** $\text{FE.Enc}(\text{MPK}, x)$: On input the master public key MPK and an input message $x \in \mathcal{X}$, do the following:
 1. For $i = 1, 2, \dots, S$, choose uniformly random $R_i \xleftarrow{\$} \{0, 1\}^r$.
 2. Run the bounded degree encryption algorithm BdFE.Enc to produce a ciphertext

$$\text{CT} \leftarrow \text{BdFE.Enc}(\text{MPK}, (x, R_1, \dots, R_S))$$

Output CT as the ciphertext.

- **Decryption** $\text{FE.Dec}(\text{SK}_C, \text{CT})$: On input a secret key SK_C and a ciphertext CT ,
 - Run the bounded FE decryption algorithm to get $\tilde{y} \leftarrow \text{BdFE.Dec}(\text{SK}_{C,\Delta}, \text{CT})$.
 - Run the randomized encoding decoder on \tilde{y} to get the output $y \leftarrow \text{RE.Decode}(\tilde{y})$.

6.0.1 Correctness

We first show correctness of the scheme \mathcal{FE} . Given a secret key SK_C and a ciphertext $\text{CT} \leftarrow \text{FE.Enc}(\text{MPK}, x)$, the decryption algorithm computes

$$\tilde{y} = \text{BdFE.Dec}(\text{SK}_{C,\Delta}, \text{CT}) = G_{C,\Delta}(x, R_1, \dots, R_S) = \tilde{C}(x; \bigoplus_{a \in \Delta} R_a)$$

Of course, running RE.Decode on this should return $y = C(x)$, by the correctness of the randomized encoding scheme.

Bootstrapping for Unbounded Queries. Although the transformation above assumes the knowledge of q (the bound on the number of secret key queries of the adversary), we can generalize it to work for unbounded queries as follows. Essentially, the idea is to generate fresh (computational) randomness for each randomized encoding using a pseudo-random function.

In particular, let $\{\text{prf}_S\}_{S \in \{0,1\}^\kappa}$ be a circuit family of weak pseudo-random functions. Consider a new circuit family \mathcal{C} that works in the following way:

$$G_{C,R}(x, S) := \tilde{C}\left(x; \text{prf}_S(R)\right)$$

Then, essentially the same construction as above works as a way to bootstrap an FE scheme for arbitrary circuits from FE schemes for circuits that can compute the weak PRF followed by the randomized encoding. Assuming the existence of weak PRFs and PRGs that can be computed by circuits of degree $\text{poly}(\kappa)$, we then obtain functional encryption schemes for arbitrary circuits. Note, that by [AGVW12] it is impossible to achieve functional encryption for PRFs under NA-SIM-security for unbounded queries. However, constructions secure under a weaker security definition (for example, indistinguishability) are still open.

6.1 Proof of Security

Theorem 6.1. *Let \mathcal{BDFE} be a q -AD-SIM-secure (resp. q -NA-SIM-secure) functional encryption scheme for any family of circuits computable in NC1. Then, for any family \mathcal{C} of polynomial-size circuits the \mathcal{FE} scheme described above is q -AD-SIM-secure (resp. q -NA-SIM-secure).*

We prove that the construction \mathcal{FE} given in Section 6 is q -AD-SIM-secure if we start out with a q -AD-SIM-secure scheme. This subsumes the non-adaptive variant of the proof.

Proof overview. Suppose the adversary sees q^* queries before seeing the ciphertext. The simulator has to simulate the ciphertext and answer the remaining secret key queries. We may again assume that the simulator knows all of $\Gamma_1, \dots, \Gamma_q, \Delta_1, \dots, \Delta_q$.

Simulating the ciphertext. The simulator gets $\{C_j(x), C_j, \text{SK}_{C_j}\}_{j \in [q^*]}$ and outputs:

$$\text{CT} \leftarrow \text{BdFE.Sim}\left(\text{MPK}, \{\text{RE.Sim}(C_j(x)), G_{C_j, \Delta_j}, \text{SK}_{C_j, \Delta_j}\}_{j \in [q^*]}\right)$$

with fresh independent randomness for each of the q^* invocations of RE.Sim.

Simulating secret key SK_{C_j} , for $j > q^*$. Here, the simulator gets MSK and $C_j(x), C_j$ and needs to simulate $\text{SK}_{C_j} := (\Delta_j, \text{SK}_{C_j, \Delta_j})$. It proceeds as follows:

1. Picks $\tilde{y}_j \leftarrow \text{RE.Sim}(C_j(x))$.
2. Runs $\text{BdFE.Sim}(\text{MSK}, (\tilde{y}_j, G_{C_j, \Delta_j}))$ to obtain $\text{SK}_{C_j, \Delta_j}$ so that CT decrypts to \tilde{y}_j .

Output $\text{SK}_{C_j} = (\Delta_j, \text{SK}_{C_j, \Delta_j})$.

Details. We establish security via a series of Games, where the last Game corresponds to the simulator described above.

Game 0. Normal encryption.

Game 1. We modify the distribution of the ciphertext to use BdFE.Sim as in the static case for both the ciphertext and the secret-key queries after the adversary sees the ciphertext. That is,

$$\text{CT} \leftarrow \text{BdFE.Sim}\left(\text{MPK}, \{G_{C_j, \Delta_j}(x; R_1, \dots, R_S), G_{C_j, \Delta_j}, \text{SK}_{C_j, \Delta_j}\}_{j \in [q^*]}\right)$$

Moreover, for $j > q^*$, it

1. Picks $\tilde{y}_j \leftarrow G_{C_j, \Delta_j}(x; R_1, \dots, R_S)$.
2. Runs $\text{BdFE.Sim}(\text{MSK}, (\tilde{y}_j, G_{C_j, \Delta_j}))$ to obtain $\text{SK}_{C_j, \Delta_j}$ so that CT decrypts to \tilde{y}_j .

Output $\text{SK}_{C_j} = (\Delta_j, \text{SK}_{C_j, \Delta_j})$.

Game 2. We replace $\{\bigoplus_{a \in \Delta_j} R_a\}_{j \in [q]}$ with $\{R'_j\}_{j \in [q]}$, where for each j :

$$G_{C_j, \Delta_j}(x; R_1, \dots, R_S) := \tilde{C}(x; \bigoplus_{a \in \Delta_j} R_a)$$

Game 3. The output of the simulator (that is, switch to using RE.Sim).

Claim 6.1.1. *The outputs of Game 0 and Game 1 are computationally indistinguishable.*

Proof. This follows readily from q -AD-SIM-security of the underlying FE scheme. □

Claim 6.1.2. *The outputs of Game 1 and Game 2 are identically distributed.*

Proof. By cover-freeness of $\Delta_1, \dots, \Delta_q$, we have that

$$\left\{ \bigoplus_{a \in \Delta_j} R_a \right\}_{j \in [q]} \quad \text{and} \quad \left\{ R'_j \right\}_{j \in [q]}$$

are identically distributed. □

Claim 6.1.3. *The outputs of Game 2 and Game 3 are computationally indistinguishable.*

Proof. This follows readily from a hybrid argument and the security of the randomized encoding scheme, which says that for each $j = 1, \dots, q$:

$$\tilde{C}_j(x; R'_j) \quad \text{and} \quad \text{RE.Sim}(C_j(x))$$

are computationally indistinguishable. □

7 Yet Another Bootstrapping Theorem Using FHE

We show a bootstrapping theorem that transforms a q -query FE scheme supporting NC1 circuits into a q -query FE scheme for arbitrary polynomial-size circuits using, in addition, a fully homomorphic encryption scheme [Gen09, BV11]. Intuitively, the construction can be viewed as follows: we reduce functional encryption for a circuit C to one for the decryption algorithm for a fully homomorphic encryption scheme computable in NC1. Putting this together with the q -query, NC1 circuit scheme from Section 5 gives us Theorem 7.1.

First, we need a generalization of the construction for NC1 circuits from Section 5. Assume that the message is split into a *public part* and a *secret part*. Then, the key observation is that the construction from Section 5 works for any circuit C which is computable in NC1 in the variables of the secret part. The rationale for this is the same as that used to obtain a predicate encryption with public index from the scheme in Section 5.

We show the following theorem:

Theorem 7.1. *Let \mathcal{BDFE} be a q -query, FE scheme which works for any NC1 circuit, and let \mathcal{FHE} be a semantically secure fully homomorphic encryption scheme whose decryption algorithm $\text{FHE.Dec}(\text{SK}, ct)$ can be implemented by an NC1 circuit in the secret key. Then, for any family of poly-size circuits \mathcal{C} there exists a q -query FE scheme $\mathcal{FE} = (\text{FE.Setup}, \text{FE.Keygen}, \text{FE.Enc}, \text{FE.Dec})$.*

Furthermore, if \mathcal{BDFE} is q -NA-SIM-secure (resp. q -AD-SIM-secure), then so is \mathcal{FE} .

Any of the recent fully homomorphic encryption schemes have decryption algorithms computable in NC1. Putting these together, we get q -bounded FE schemes under the “learning with errors” and the “ring learning with errors” assumptions (together with certain circular security assumptions) [BV11].

Let \mathcal{C} be an arbitrary polynomial-size circuit family. Our construction uses the following components:

- **An Inner Encryption Scheme:** Let $\mathcal{FHE} = (\text{FHE.Keygen}, \text{FHE.Enc}, \text{FHE.Eval}, \text{FHE.Dec})$ be a fully homomorphic encryption scheme where the decryption algorithm FHE.Dec can be implemented by an NC1 circuit in the secret key.
- **An Outer Encryption Scheme:** Let $\mathcal{BDFE} = (\text{BdFE.Setup}, \text{BdFE.Keygen}, \text{BdFE.Enc}, \text{BdFE.Dec})$ be a q -query functional encryption scheme for the family \mathcal{G} that is computable by NC1 circuits in their secret input defined as follows:

$$G_C(ct, \text{SK}) := [ct, \text{FHE.Dec}(\text{SK}, \text{FHE.Eval}(C, ct))]$$

Note that although \mathcal{G} has circuits that are at least as large as those for \mathcal{C} , all we are interested in is its degree in the secret input, namely SK .

Our q -query secure encryption scheme $(\text{FE.Setup}, \text{FE.Keygen}, \text{FE.Enc}, \text{FE.Dec})$ for \mathcal{C} works as follows.

- **Setup** $\text{FE.Setup}(1^\kappa)$: Run the bounded FE setup algorithm to generate a master public-key/secret-key pair:

$$(\text{MPK}, \text{MSK}) \leftarrow \text{BdFE.Setup}(1^\kappa)$$

- **Key Generation** $\text{FE.Keygen}(\text{MSK}, C)$: On input the master secret key MSK and a circuit $C \in \mathcal{C}$, run the bounded FE key generation algorithm to generate a secret key

$$\text{SK}_C \leftarrow \text{BdFE.Keygen}(\text{MSK}, G_C)$$

for the circuit G_C .

- **Encryption** $\text{FE.Enc}(\text{MPK}, x)$: On input the master public key MPK and an input message $x \in \mathcal{X}$:

1. Choose a uniformly random public-key/secret-key pair for the fully homomorphic encryption scheme \mathcal{FHE} by running

$$(\text{PK}, \text{SK}) \leftarrow \text{FHE.Keygen}(1^\kappa)$$

2. Encrypt the input message x using the FHE encryption algorithm

$$ct \leftarrow \text{FHE.Enc}(\text{PK}, x)$$

3. Run the bounded FE encryption algorithm to encrypt the ciphertext ct together with the fully homomorphic secret key SK :

$$\text{CT} \leftarrow \text{BdFE.Enc}(\text{MPK}, (ct, \text{SK}))$$

Output CT as the ciphertext.

- **Decryption** $\text{FE.Dec}(\text{SK}_C, \text{CT})$: On input a secret key SK_C and a ciphertext CT , run the bounded FE decryption algorithm to get $[ct, y] \leftarrow \text{BdFE.Dec}(\text{SK}_C, \text{CT})$, and output $[ct, y]$.

7.0.1 Correctness and Security

We first show correctness of the scheme \mathcal{FE} . Given a secret key SK_C and a ciphertext $\text{CT} \leftarrow \text{FE.Enc}(\text{MPK}, x)$, the decryption algorithm computes

$$\begin{aligned} [ct, y] &= \text{BdFE.Dec}(\text{SK}_C, \text{CT}) \\ &= \text{BdFE.Dec}(\text{SK}_C, \text{BdFE.Enc}(\text{MPK}, (ct, \text{SK}))) \\ &\quad (\text{where } ct \leftarrow \text{FHE.Enc}(\text{PK}, x)) \\ &= G_C(ct, \text{SK}) \\ &= [ct, \text{FHE.Dec}(\text{SK}, \text{FHE.Eval}(C, ct))] \\ &= [ct, C(x)] \end{aligned}$$

We establish security via a series of Games. The simulator is described in Game 2.

Game 0. Normal encryption.

Game 1. Run the q -query simulator on input $([ct \leftarrow \text{FHE.Enc}(\text{PK}, x), C_i(x)], G_{C_i}, \text{SK}_i)_{i=1}^n$, where $n \leq q$ is the number of oracle query calls made to BdFE.Keygen .

Game 2. Run the q -query simulator on input $([ct \leftarrow \text{FHE.Enc}(\text{PK}, 0), C_i(x)], G_{C_i}, \text{SK}_i)_{i=1}^n$, where $n \leq q$ is the number of oracle query calls made to BdFE.Keygen .

References

- [AGVW12] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. Cryptology ePrint Archive, Report 2012/468, 2012. <http://eprint.iacr.org/>.
- [AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Computational Complexity*, 15(2):115–162, 2006.
- [BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.
- [BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC ’88, pages 1–10, New York, NY, USA, 1988. ACM.
- [BMR90] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, pages 503–513, 1990.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.
- [BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.
- [CFGN96] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *STOC*, pages 639–648, 1996. Longer version at <http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TR-682.pdf>.
- [CHH⁺07] Ronald Cramer, Goichiro Hanaoka, Dennis Hofheinz, Hideki Imai, Eike Kiltz, Rafael Pass, Abhi Shelat, and Vinod Vaikuntanathan. Bounded CCA2-secure encryption. In *ASIACRYPT*, pages 502–518, 2007.
- [Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.
- [DI05] Ivan Damgård and Yuval Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, pages 378–394, 2005.
- [DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In *In EUROCRYPT*, pages 65–82. Springer-Verlag, 2002.

- [DN00] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In *CRYPTO*, pages 432–450, 2000.
- [Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009. crypto.stanford.edu/craig.
- [GLW12] Shafi Goldwasser, Allison B. Lewko, and David A. Wilson. Bounded-collusion IBE from key homomorphism. In *TCC*, pages 564–581, 2012.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.
- [IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *FOCS*, pages 294–304, 2000.
- [IKOS07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, STOC '07, pages 21–30, New York, NY, USA, 2007. ACM.
- [KO04] Jonathan Katz and Rafail Ostrovsky. Round-optimal secure two-party computation. In *CRYPTO*, pages 335–354, 2004.
- [KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.
- [LOS⁺10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 62–91. Springer, 2010.
- [O’N10] Adam O’Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. <http://eprint.iacr.org/>.
- [OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.
- [OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 191–208. Springer, 2010.
- [Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22:612–613, November 1979.
- [Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

- [SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM Conference on Computer and Communications Security*, pages 463–472, 2010.
- [SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 457–473. Springer, 2009.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
- [Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

A Relations between Definitions of Functional Encryption

In this section, we first describe simulation-based and indistinguishability-based definitions for many input messages functional encryption, largely based on the recent works of Boneh, Sahai and Waters [BSW11] and O’Neill [O’N10]. We then go on to show relations between various flavors of these definitions.

A.1 A Simulation-based Definition

Definition A.1 (NA-SIM- and AD-SIM- Security). *Let \mathcal{FE} be a functional encryption scheme for a circuit family $\mathcal{C} = \{\mathcal{C}_\kappa : \mathcal{X}_\kappa \rightarrow \mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$. For every p.p.t. adversary $A = (A_1, A_2)$ and a p.p.t. simulator $S = (S_1, S_2)$, consider the following two experiments:*

$\text{Exp}_{\mathcal{FE}, \ell, A}^{\text{real}}(1^\kappa):$	$\text{Exp}_{\mathcal{FE}, \ell, S}^{\text{ideal}}(1^\kappa):$
1: $(\text{MPK}, \text{MSK}) \leftarrow \text{FE.Setup}(1^\kappa)$	1: $(\text{MPK}, \text{MSK}) \leftarrow \text{FE.Setup}(1^\kappa)$
2: $(x_1, \dots, x_\ell, st) \leftarrow A_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$	2: $(x_1, \dots, x_\ell, st) \leftarrow A_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$
	► Let (C_1, \dots, C_q) be A_1 ’s oracle queries
	► Let SK_i be the oracle reply to C_i
	► Let $\mathcal{V} := \{y_{ij} = C_i(x_j), C_i, \text{SK}_i\}$.
3: $\text{CT}_i \leftarrow \text{FE.Enc}(\text{MPK}, x_i)$	3: $(\text{CT}_1, \dots, \text{CT}_\ell, st') \leftarrow S_1(\text{MPK}, \mathcal{V}, 1^{ x_i })$
4: $\alpha \leftarrow A_2^{\mathcal{O}(\text{MSK}, \cdot)}(\text{MPK}, \text{CT}_1, \dots, \text{CT}_\ell, st)$	4: $\alpha \leftarrow A_2^{\mathcal{O}'(\text{MSK}, st', \cdot)}(\text{MPK}, \text{CT}_1, \dots, \text{CT}_\ell, st)$
5: <i>Output</i> $(\alpha, x_1, \dots, x_\ell)$	5: <i>Output</i> $(\alpha, x_1, \dots, x_\ell)$

We distinguish between two cases of the above experiment:

1. The adaptive case, where:
 - the oracle $\mathcal{O}(\text{MSK}, \cdot) = \text{FE.Keygen}(\text{MSK}, \cdot)$ and

- the oracle $\mathcal{O}'(\text{MSK}, st', \cdot)$ is the second stage of the simulator, namely $S_2^{U_x(\cdot)}(\text{MSK}, st', \cdot)$, where $U_x(C) = C(x)$ for any $C \in \mathcal{C}$.

The simulator algorithm S_2 is stateful in that after each invocation, it updates the state st' which is carried over to its next invocation. We call a simulator algorithm $S = (S_1, S_2)$ admissible if, on each input C , S_2 makes just a single query to its oracle $U_x(\cdot)$ on C itself.

The functional encryption scheme \mathcal{FE} is then said to be (q, many) -simulation-secure for many messages against adaptive adversaries $((q, \text{many})\text{-AD-SIM-secure}$, for short) if there is an admissible p.p.t. simulator $S = (S_1, S_2)$ such that for every polynomial function $\ell = \ell(\kappa)$ and for every p.p.t. adversary $A = (A_1, A_2)$ that makes at most q queries, the following two distributions are computationally indistinguishable:

$$\left\{ \text{Exp}_{\mathcal{FE}, A}^{\text{real}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx} \left\{ \text{Exp}_{\mathcal{FE}, S}^{\text{ideal}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}}$$

In the special case where $\ell(\kappa) = 1$, we will call the scheme $(q, \text{one})\text{-AD-SIM-secure}$.

2. The non-adaptive case, where the oracles $\mathcal{O}(\text{MSK}, \cdot)$ and $\mathcal{O}'(\text{MSK}, st, \cdot)$ are both the “empty oracles” that return nothing: the functional encryption scheme \mathcal{FE} is then said to be (q, many) -query simulation-secure for many messages against non-adaptive adversaries $((q, \text{many})\text{-NA-SIM-secure}$, for short) if there is a p.p.t. simulator $S = (S_1, \perp)$ such that for every polynomial function $\ell = \ell(\kappa)$ for every p.p.t. adversary $A = (A_1, A_2)$ that makes at most q queries, the two distributions above are computationally indistinguishable. In the special case where $\ell(\kappa) = 1$, we will call the scheme $(q, \text{one})\text{-NA-SIM-secure}$.

Note that this definition is the generalization of the one presented in Section 3 to the case where the adversary receives multiple ciphertexts. Intuitively, the above security definition states that whatever information adversary is able to learn from the ciphertexts and secret keys, can be obtained by a simulator from the secret keys and the outputs of the functionality for the messages only.

We remark that our definitions imply (and are stronger than) those of presented in the work of Boneh, Sahai and Waters [BSW11]. More formally, for the adaptive variant we can instantiate [BSW11] simulator $(\text{Sim}_1, \text{Sim}_O, \text{Sim}_2)$ as follows.

1. Sim_1 runs FE.Setup and sets $pp := \text{MPK}, \sigma := \text{MSK}$.
2. Sim_O runs FE.Keygen algorithm on MSK and updates σ to include all oracle queries and replies (C_i, SK_i) .
3. Sim_2 computes $y_i = U_x(\cdot)$ for all C_i using its oracle. Next, it runs our simulator $S_1(\text{MPK}, \{y_i, C_i, \text{SK}_i\})$ to obtain the ciphertext CT . It invokes A° on the ciphertext, and on any FE.Keygen call it uses our S_2 to obtain a secret key. Finally, output the same α as A° . The non-adaptive variant follows similarly.

A.2 An Indistinguishability-Based Definition

Definition A.2 (NA-IND- and AD-IND-Security). Let \mathcal{FE} be a functional encryption scheme for a circuit family $\mathcal{C} = \{C_\kappa : \mathcal{X}_\kappa \rightarrow \mathcal{Y}_\kappa\}_{\kappa \in \mathbb{N}}$. For every function $\ell = \ell(\kappa)$, every p.p.t. adversary $A = (A_1, A_2)$, consider the following two experiments:

$$\text{Exp}_{\mathcal{FE},A}^{(0)}(1^\kappa):$$

- 1: $(\text{MPK}, \text{MSK}) \leftarrow \text{FE.Setup}(1^\kappa)$
 - 2: $(\vec{x}_0, \vec{x}_1, st) \leftarrow A_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$
 - where $\vec{x}_0 = (x_0[1], \dots, x_0[\ell])$
 - and $\vec{x}_1 = (x_1[1], \dots, x_1[\ell])$
 - 3: $\boxed{\text{CT}_i \leftarrow \text{FE.Enc}(\text{MPK}, x_0[i]) \quad \forall i \in [\ell]}$
 - 4: $b \leftarrow A_2^{\mathcal{O}(\text{MSK}, \cdot)}(\text{MPK}, \text{CT}_1, \dots, \text{CT}_\ell, st)$
 - 5: Output b
-

$$\text{Exp}_{\mathcal{FE},A}^{(1)}(1^\kappa):$$

- 1: $(\text{MPK}, \text{MSK}) \leftarrow \text{FE.Setup}(1^\kappa)$
 - 2: $(\vec{x}_0, \vec{x}_1, st) \leftarrow A_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$
 - where $\vec{x}_0 = (x_0[1], \dots, x_0[\ell])$
 - and $\vec{x}_1 = (x_1[1], \dots, x_1[\ell])$
 - 3: $\boxed{\text{CT}_i \leftarrow \text{FE.Enc}(\text{MPK}, x_1[i]) \quad \forall i \in [\ell]}$
 - 4: $b \leftarrow A_2^{\mathcal{O}(\text{MSK}, \cdot)}(\text{MPK}, \text{CT}_1, \dots, \text{CT}_\ell, st)$
 - 5: Output b
-

Define an *admissible adversary* $A = (A_1, A_2)$ as one which makes at most q oracle queries and $C(x_0[i]) = C(x_1[i])$ for each query C and every $i \in [\ell]$. We distinguish between two cases of the above experiment:

1. The adaptive case, where the oracle $\mathcal{O}(\text{MSK}, \cdot) = \text{FE.Keygen}(\text{MSK}, \cdot)$: the functional encryption scheme \mathcal{FE} is said to be *indistinguishable-secure for many messages against adaptive adversaries* ((q, many) -AD-IND-secure, for short) if for every polynomial function $\ell = \ell(\kappa)$ and every admissible p.p.t. admissible adversary $A = (A_1, A_2)$, the advantage of A defined as below is negligible in the security parameter κ :

$$\text{Adv}_{\mathcal{FE}, \ell, A}(\kappa) := |\Pr[\text{Exp}_{\mathcal{FE}, \ell, A}^{(0)}(1^\kappa) = 1] - \Pr[\text{Exp}_{\mathcal{FE}, \ell, A}^{(1)}(1^\kappa) = 1]|$$

where the probability is over the random coins of the algorithms of the scheme \mathcal{FE} and that of A . In the special case where $\ell(\kappa) = 1$, we will call the scheme (q, one) -AD-IND-secure.

2. The non-adaptive case, where the oracle $\mathcal{O}(\text{MSK}, \cdot)$ is the “empty oracle” that returns nothing: the functional encryption scheme \mathcal{FE} is said to be *indistinguishable-secure for many messages against non-adaptive adversaries* ((q, many) -NA-IND-secure, for short) if for every polynomial function $\ell = \ell(\kappa)$ and every admissible p.p.t. adversary $A = (A_1, A_2)$, the advantage of A defined as above is negligible in the security parameter κ .

In the special case where $\ell(\kappa) = 1$, we will call the scheme (q, one) -NA-IND-secure.

Note that this definition is identical to the definitions presented in [BSW11] and [O’N10], except that they define it for a single message only.

A.3 Relations Between Definitions

In this section, we prove the following relations between the definitions.

- *Non-adaptive Definitions:* When considering non-adaptive definitions (namely, where the adversary is constrained to making secret key queries only *before* he receives the challenge ciphertext), we show that *one-message* definitions are equivalent to *many-message* definitions, both in the indistinguishability and the simulation worlds.

Put together, this shows that it is sufficient to prove security for one message in the simulation sense, which is precisely what we will do for our schemes.

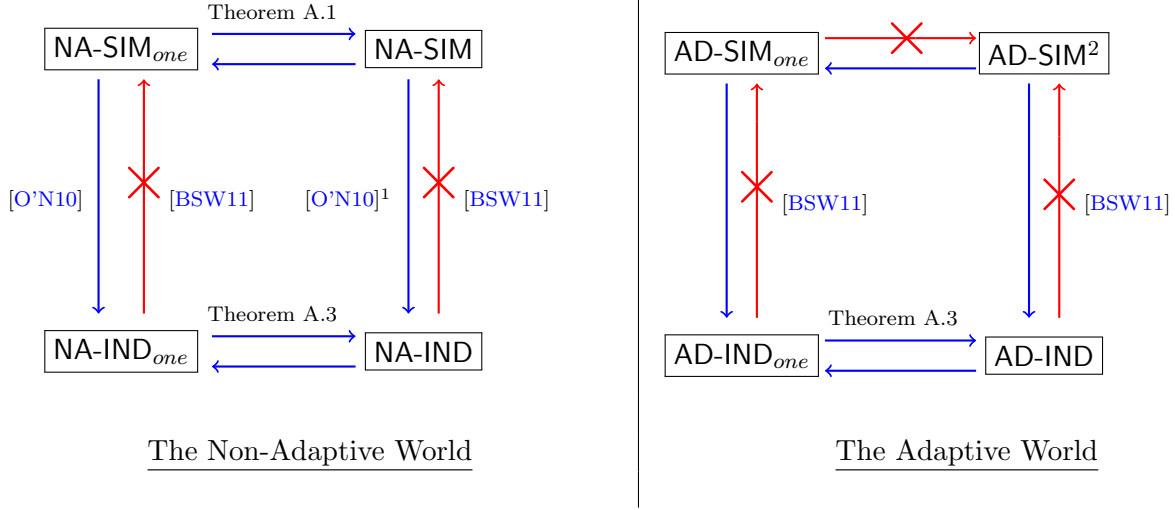


Figure 1: Relations between definitions of functional encryption in the non-adaptive and adaptive flavors. Regular blue arrows indicate an implication between the definitions, and a red arrow with a cross on it indicates a separation. The citations above (i.e. $\text{AD-SIM} = (q, \text{many})\text{-AD-SIM}$, $\text{AD-SIM}_{\text{one}} = (q, \text{one})\text{-AD-SIM}$; similarly for the rest of the abbreviations.)

- *Adaptive Definitions:* When considering adaptive definitions (namely, where the adversary is allowed to make secret key queries *after* receiving the challenge ciphertext) we show that for any q , $(q, \text{one})\text{-AD-SIM}$ implies $(q, \text{one})\text{-AD-IND}$ which is equivalent to $(q, \text{many})\text{-AD-IND}$. We also construct a functional encryption scheme and prove it secure under $(q, \text{one})\text{-AD-SIM}$ definition. Therefore, from the work of Boneh et al. [BSW11] we can conclude that $(q, \text{one})\text{-AD-SIM}$ does not imply $(q, \text{many})\text{-AD-SIM}$.

These relationships are summarized in Figure 1.

Theorem A.1. *Let \mathcal{FE} be $(q, \text{one})\text{-NA-SIM}$ -secure functional encryption scheme for a circuit family \mathcal{C} . Then, \mathcal{FE} is also $(q, \text{many})\text{-NA-SIM}$ -secure.*

Proof. Let S_1 be the single message p.p.t. simulator. We construct a p.p.t. simulator S_m . Intuitively, the multiple message simulator will just invoke the single message simulator many times. Then, using the standard hybrid argument we can conclude that it produces output indistinguishable from the real. Let $\ell = \ell(k)$ be arbitrary polynomial function and let $A = (A_1, A_2)$ be arbitrary p.p.t. adversary.

On input $(\text{MPK}, \{y_{ij} = C_i(x_j), C_i, \text{SK}_i\})$ the simulator S_m proceeds as follows: For each j , let

$$V_j := \{y_{ij} = C_i(x_j), C_i, \text{SK}_i\}_{i \in [q]}$$

¹This proof was not explicitly given in [O'N10], but a similar proof for single message definitions can be easily extended.

²General functional encryption for this definition was shown impossible in [BSW11] when adversary makes just 2 FE.Keygen calls (2-bounded collusion). Since we show a secure construction satisfying $\text{AD-SIM}_{\text{one}}$, this implication follows.

The simulator computes and outputs the ciphertext¹:

$$(\text{CT}_1, \dots, \text{CT}_\ell), \text{ where } \text{CT}_j \leftarrow S_1(\text{MPK}, V_j)$$

Now, let D be the distinguisher between the real and ideal experiments. Then, by the hybrid argument D can distinguish between the experiments where A_2 is given

$$(\text{CT}_1^r, \dots, \text{CT}_{i-1}^r, \text{CT}_i^s, \dots, \text{CT}_\ell^s) \text{ vs } (\text{CT}_1^r, \dots, \text{CT}_i^r, \text{CT}_{i+1}^s, \dots, \text{CT}_\ell^s)$$

for some i , where CT^r 's and CT^s 's correspond to the real and simulated ciphertexts, respectively.

We now construct a single message adversary $B = (B_1, B_2)$ and a distinguisher D' as follows:

1. $B_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$ runs A_1 and replies to its oracle queries appropriately to get (x_1, \dots, x_ℓ, st) . It outputs

$$(x_i, st' = (x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_\ell, st, (C_j, \text{SK}_j)_{j \in [q]}))$$

2. $B_2(\text{MPK}, \text{CT}, st')$ first runs the real encryption algorithm on input messages x_1, \dots, x_{i-1} to obtain $\text{CT}_1^r, \dots, \text{CT}_{i-1}^r$. Then, for all $j \geq i + 1$ it sets

$$V_j := \{y_{ij} = C_i(x_j), C_i, \text{SK}_i\}_{i \in [q]}$$

and runs the single message simulator to get a ciphertext $\text{CT}_j^s \leftarrow S_1(\text{MPK}, V_j)$.

3. Finally, it invokes $A_2(\text{MPK}, \text{CT}_1^r, \dots, \text{CT}_{i-1}^r, \text{CT}, \text{CT}_{i+1}^s, \dots, \text{CT}_\ell^s)$ and outputs whatever it outputs.
4. The distinguisher D' is the same as D .

We showed that if there exists a distinguisher for many message simulator, then we can break the security for the single message simulator. This concludes the proof. \square

Theorem A.2. *Let \mathcal{FE} be (q, one) -AD-SIM-secure functional encryption scheme for a circuit family \mathcal{C} . Then, \mathcal{FE} is also (q, one) -AD-IND-secure.*

Proof. Let $A = (A_1, A_2)$ be the admissible adversary such that $\text{Adv}_{\mathcal{FE}, \ell, A}$ is non-negligible. We construct adversary $B = (B_1, B_2)$ against (q, one) -AD-SIM-security.

- $B_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{MPK})$: Run the adversary A_1 and reply to its oracle queries using its own oracle to obtain (x_0, x_1, st) . Output $(x_b, st^* := (st, x_0, x_1))$, where $b \xleftarrow{\$} \{0, 1\}$.
- $B_2^{\mathcal{O}'(\text{MSK}, st', \cdot)}(\text{MPK}, \text{CT}, st)$: Run the adversary $A_2(\text{MPK}, \text{CT}, st)$ replying to its oracle queries using its own oracle to obtain b' . Output $\alpha := (b', st')$.

¹Note, that this theorem does not extend to the adaptive definition. In particular, the proof breaks down when even trying to construct the multiple message simulator to “forge” the secret keys SK .

Now, in the real experiment $b = b'$ with probability $1/2 + \epsilon$ for some noticeable ϵ . In the ideal experiment since the simulator is admissible, it must make the same oracle queries to $U_x(\cdot)$ as B_2 makes, which are the same queries as A_2 makes. Hence, it must be the case that $C_j(x_0) = C_j(x_1)$ for all j . Therefore, information theoretically the simulator gets no information about the bit b and hence cannot produce the corresponding ciphertext with probability better than $1/2$. Hence, we can distinguish between the ideal and real experiment. \square

Theorem A.3. *Let \mathcal{FE} be (q, one) -AD-IND/NA-IND-secure functional encryption scheme for a circuit family \mathcal{C} . Then, \mathcal{FE} is also (q, many) -AD-IND/NA-IND-secure, respectively.*

Proof. These proofs follow a standard hybrid argument. \square

As a result, we focus on proving only (q, one) -NA-SIM and (q, one) -AD-SIM for our constructions. For simplicity we denote it as q -NA-SIM- and q -AD-SIM- security.

B Probabilistic Proofs

B.1 Small Pairwise Intersection

Lemma B.1. *Let $\Gamma_1, \dots, \Gamma_q \subseteq [N]$ be randomly chosen subsets of size $tD + 1$. Let $t = \Theta(q^2\kappa)$, $N = \Theta(D^2q^2t)$. Then,*

$$\Pr\left[\left|\bigcup_{i \neq j} (\Gamma_i \cap \Gamma_j)\right| \leq t\right] = 1 - 2^{-\Omega(\kappa)}$$

where the probability is over the random choice of the subsets $\Gamma_1, \dots, \Gamma_q$.

Proof. For all $i, j \in [q]$ such that $i \neq j$, let X_{ij} be a random variable denoting the size of the intersection of S_i and S_j . Let

$$X = \sum_{i, j \in [q], i \neq j} X_{ij}$$

It is not hard to see that X_{ij} 's are independent random variables. By the linearity of expectation,

$$E[X] = \sum_{i, j \in [q], i \neq j} E[X_{ij}]$$

Now, for a fixed set S_i and a randomly chosen S_j the size of the intersection of S_i and S_j follows a hypergeometric distribution, where $tD + 1$ serves both as the number of success states and number of trials, and N is the population size. Therefore,

$$E[X_{ij}] = \frac{(tD + 1)(tD + 1)}{N} = \frac{(tD + 1)^2}{N}$$

Hence,

$$\mu = E[X] = \frac{q(q - 1)(tD + 1)^2}{N} \leq \frac{10q^2t^2D^2}{N}$$

By Chernoff bound, for any $\sigma \geq 0$:

$$\Pr[X > (1 + \sigma)\mu] < \exp\left(\frac{-\sigma^2}{2 + \sigma}\mu\right)$$

Setting $t = \Theta(q^2\kappa)$, $N = \Theta(D^2q^2t)$ gives us $\mu = \Theta(t) = \Theta(q^2\kappa)$. Applying Chernoff bound,

$$\Pr[X > t] = 2^{-\Omega(\kappa)}$$

□

B.2 Cover-Freeness

Lemma B.2. *Let $\Delta_1, \dots, \Delta_q \subseteq [S]$ be randomly chosen subsets of size v . Let $v(\kappa) = \Theta(\kappa)$ and $S(\kappa) = \Theta(vq^2)$. Then, for all $i \in [q]$*

$$\Pr[\Delta_i \setminus \left(\bigcup_{j \neq i} \Delta_j\right) \neq \emptyset] = 1 - 2^{-\Omega(\kappa)}$$

where the probability is over the random choice of subsets $\Delta_1, \dots, \Delta_q$.

Proof. Let $i \in [q]$ be arbitrary. Let $G := \bigcup_{j \neq i} \Delta_j$. Clearly, $|G| = (q-1)v$. Let X be the random variable denoting $|\Delta_i \setminus G|$. Now,

$$|\Delta_i \setminus G| = |\Delta_i| - |\Delta_i \cap G| = v - |\Delta_i \cap G|$$

Hence,

$$E[X] = v - E[|\Delta_i \cap G|]$$

Now, $E[|\Delta_i \cap G|]$ follows a hypergeometric distribution with v success states, $v(q-1)$ trials and S population size. Hence,

$$E[|\Delta_i \cap G|] = \frac{v^2(q-1)}{S}$$

Therefore, $E[X] = v - (v^2(q-1))/S$. Setting, $v(\kappa) = \Theta(\kappa)$ and $S(\kappa) = \Theta(vq^2)$ we obtain that $\mu = E[X] = \Theta(\kappa)$. By Chernoff bound, for any $0 \leq \sigma \leq 1$:

$$\Pr[X \leq (1-\sigma)\mu] < \exp\left(\frac{-\sigma^2}{2}\mu\right)$$

Applying it we obtain that $\Pr[X \leq (1-\sigma)\mu] = 2^{-\Omega(\kappa)}$. Hence,

$$\Pr[\Delta_i \setminus \left(\bigcup_{j \neq i} \Delta_j\right) \neq \emptyset] = \Pr[X > 0] \geq \Pr[X > (1-\sigma)\mu] = 1 - 2^{-\Omega(\kappa)}$$

□

Attribute-Based Encryption for Circuits

Sergey Gorbunov*

Vinod Vaikuntanathan[†]

Hoeteck Wee[‡]

May 31, 2013

Abstract

In an attribute-based encryption (ABE) scheme, a ciphertext is associated with an ℓ -bit *public index* ind and a message m , and a secret key is associated with a Boolean predicate P . The secret key allows to decrypt the ciphertext and learn m iff $P(\text{ind}) = 1$. Moreover, the scheme should be secure against collusions of users, namely, given secret keys for polynomially many predicates, an adversary learns nothing about the message if none of the secret keys can individually decrypt the ciphertext.

We present attribute-based encryption schemes for circuits of any arbitrary polynomial size, where the public parameters and the ciphertext grow linearly with the depth of the circuit. Our construction is secure under the standard learning with errors (LWE) assumption. Previous constructions of attribute-based encryption were for Boolean formulas, captured by the complexity class NC^1 .

In the course of our construction, we present a new framework for constructing ABE schemes. As a by-product of our framework, we obtain ABE schemes for polynomial-size branching programs, corresponding to the complexity class $LOGSPACE$, under quantitatively better assumptions.

*University of Toronto. Email: sgorbunov@cs.toronto.edu. Supported by Ontario Graduate Scholarship (OGS).

[†]University of Toronto. Email: vinodv@cs.toronto.edu. Supported by an NSERC Discovery Grant and by DARPA under Agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

[‡]George Washington University. Email: hoeteck@alum.mit.edu. Supported by NSF CAREER Award CNS-1237429.

1 Introduction

Attribute-based encryption [SW05, GPSW06] is an emerging paradigm for public-key encryption which enables fine-grained control of access to encrypted data. In traditional public-key encryption, access to the encrypted data is all or nothing: given the secret key, one can decrypt and read the entire message, but without it, nothing about the message is revealed (other than its length). In attribute-based encryption, an encryption of a message m is labeled with a *public* attribute vector ind (also called the “index”), and secret keys are associated with predicates P . A secret key sk_P decrypts the ciphertext and recovers the message m if and only if ind satisfies the predicate, namely if and only if $P(\text{ind}) = 1$.

Attribute-based encryption captures as a special case previous cryptographic notions such as identity-based encryption (IBE) [Sha84, BF01, Coc01] and fuzzy IBE [SW05]. It has also found applications in scenarios that demand complex policies to control access to encrypted data, as well as in designing cryptographic protocols for verifiably outsourcing computations [PRV12].

The crucial component in the security requirement for attribute-based encryption stipulates that it resists *collusion attacks*, namely any group of users collectively learns nothing about the message m if none of them is individually authorized to decrypt the ciphertext.

In the past few years, there has been significant progress in attribute-based encryption in terms of efficiency, security guarantees, and diversifying security assumptions [GPSW06, Wat09, LW10, LOS⁺10, CHKP12, ABB10a, OT10]. On the other hand, little progress has been made in terms of supporting larger classes of predicates. The state of the art is Boolean formulas [GPSW06, LOS⁺10, OT10], which is a subclass of log-space computations. Constructing a secure attribute-based encryption for all polynomial-time predicates was posed as a central challenge by Boneh, Sahai and Waters [BSW11]. We resolve this problem affirmatively in this work.

2 Our Contributions

We construct attribute-based encryption schemes for circuits of every a-priori bounded depth, based on the learning with errors (LWE) assumption. In the course of our construction, we present a new framework for constructing attribute-based encryption schemes, based on a primitive that we call “two-to-one recoding” (TOR). Our methodology departs significantly from the current line of work on attribute-based encryption [GPSW06, LOS⁺10] and instead, builds upon the connection to garbled circuits developed in the context of *bounded* collusions [SS10b, GVV12]. Along the way, we make the first substantial progress towards the 25-year-old open problem of constructing (fully) reusable garbled circuits. In a follow-up work, Goldwasser et al. [GKP⁺13] completely resolved this open problem; moreover, their construction relies crucially on our ABE scheme as an intermediate building block. More details follow.

2.1 Attribute-based encryption

For every class of predicate circuits with depth bounded by a polynomial function $d = d(\lambda)$ (where λ is the security parameter), we construct an ABE scheme that supports this class of circuits, under the learning with errors (LWE) assumption. Informally, the (decisional) LWE problem [Reg09] asks to distinguish between “noisy” random linear combinations of n numbers $\mathbf{s} = (s_1, \dots, s_n) \in \mathbb{Z}_q^n$ from uniformly random numbers over \mathbb{Z}_q .

Regev [Reg09] showed that solving the LWE problem *on the average* is as hard as (quantumly) solving several notoriously difficult lattice problems *in the worst case*. Since then, the LWE assumption has become a central fixture in cryptography. We now have a large body of work building cryptographic schemes under the LWE assumption, culminating in the construction of a fully homomorphic encryption scheme [BV11].

The key parameter that determines the hardness of LWE is the ratio between the modulus q and the maximum absolute value of the noise B ; as such, we refer to q/B as the hardness factor of LWE. The problem becomes easier as this ratio grows, but is believed to be hard for 2^{n^ϵ} -time algorithms when $q/B = 2^{O(n^\epsilon)}$, where $0 < \epsilon < 1/2$. Our results will hold as long as the latter holds for *some constant* ϵ .

In particular, we show:

Theorem 2.1 (informal). *Assume that there is a constant $0 < \epsilon < 1$ for which the LWE problem is hard for a $\exp(n^\epsilon)$ factor in dimension n , for all large enough n . Then, for any polynomial d , there is a selectively secure attribute encryption scheme for general circuits of depth d .*

Moreover, our scheme has succinct ciphertexts, in the sense that the ciphertext size depends polynomially on the depth d and the length ℓ of the attribute vector ind , but not on the size of the circuits in the class. The construction as stated achieves the weaker notion of selective security, but we can easily obtain a fully secure scheme following [BB04] (but using sub-exponential hardness in a crucial way):

Corollary 2.2. *Assume that there is a constant $0 < \epsilon < 1/2$ such that the LWE problem with a factor of $\exp(n^\epsilon)$ is hard in dimension n for $\exp(n^\epsilon)$ -time algorithms. Then, for any polynomial d , there is a fully secure attribute-based encryption scheme for general circuits of depth d .*

We also obtain a new ABE scheme for branching programs (which correspond to the complexity class *LOGSPACE*) under the weaker quasi-polynomial hardness of LWE:

Theorem 2.3 (informal). *There exist attribute-based encryption schemes for the class of branching programs under either (1) the hardness of the LWE problem with an $n^{\omega(1)}$ factor, or (2) the bilinear decisional Diffie-Hellman assumption.*

Here, there is no a-priori bound on the size or the depth of the branching program. In addition, we achieve succinct ciphertexts of size $O(\ell)$ where ℓ is the number of bits in the index. Prior to this work, we only knew how to realize IBE and inner product encryption under $n^{\omega(1)}$ -hardness of LWE [CHKP12, ABB10a, AFBV11], whereas our bilinear construction is a different way to achieve the results of Goyal et al. [GPSW06] which uses secret-sharing for general access structures. Our construction exploits a combinatorial property of branching programs to overcome limitations of previous approaches based on secret sharing for monotone formulas (c.f. [ABV⁺12]). The construction is inspired by a pairings-based scheme for regular languages in [Wat12].

We now move on to provide a technical roadmap of our construction: first, we define a new primitive that we call a *two-to-one recoding* (TOR) scheme; we then show how TOR gives us an attribute-based encryption scheme for circuits, and how to construct a TOR scheme from the LWE assumption.

2.2 New Framework: TOR

A Two-to-One Recoding (TOR) scheme is a family of (probabilistic) functions $\{\text{Encode}(\text{pk}, \cdot)\}$ indexed by pk , together with a “two-to-one” recoding mechanism. The basic computational security guarantee for $\text{Encode}(\text{pk}, \cdot)$ is that of (correlated) pseudorandomness [RS10]: $\text{Encode}(\text{pk}, s)$ should be pseudorandom given $\text{Encode}(\text{pk}_i, s)$ for polynomially many pk_i ’s, where s is a uniformly random “seed”.

The recoding mechanism guarantees that given any triple of public keys $(\text{pk}_0, \text{pk}_1, \text{pk}_{\text{tgt}})$, there is a recoding key rk that allows us to perform the transformation

$$(\text{Encode}(\text{pk}_0, s), \text{Encode}(\text{pk}_1, s)) \mapsto \text{Encode}(\text{pk}_{\text{tgt}}, s).$$

Such a recoding key rk can be generated using either of the two secret keys sk_0 or sk_1 . Furthermore, the recoding mechanism must satisfy a natural simulation requirement: namely, we can generate rk given just pk_0, pk_1 (and neither of the two secret keys), if we are allowed to “program” pk_{tgt} . That is, there are three ways of generating the pair $(\text{pk}_{\text{tgt}}, \text{rk})$ that are (statistically) indistinguishable: (1) given pk_{tgt} , generate rk using the secret key sk_0 ; (2) given pk_{tgt} , generate rk using the secret key sk_1 ; and (3) generate rk without either secret key, by “programming” the output public key pk_{tgt} .

This requirement demonstrates the *intuitive guarantee* that we expect from a two-to-one recoding mechanism: namely, the recoding key is “useless” given only one encoding, but not both encodings. For example, it is easy to see that given $\text{Encode}(\text{pk}_0, s)$ and rk (but not $\text{Encode}(\text{pk}_1, s)$), the output $\text{Encode}(\text{pk}_{\text{tgt}}, s)$ is pseudorandom. Indeed, this is because rk could as well have been “simulated” using sk_1 , in which case it is of no help in the distinguishing task.

The simulation requirement also rules out the trivial construction from trapdoor functions where rk is a trapdoor for inverting $\text{Encode}(\text{pk}_0, \cdot)$ or $\text{Encode}(\text{pk}_1, \cdot)$.

From TOR to Garbled Circuits. We start from the observation that our TOR primitive implies a form of *reusable* garbled circuits *with no input or circuit privacy*, but instead, with a form of *authenticity guarantee*. As we will see, this leads directly into our attribute-based encryption scheme.

Consider a two-input boolean gate with input wires u, v and output wire w , computing a function $G : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$. In Yao’s garbled circuit construction, we associate each wire with a pair of strings (called “labels”), and we provide a translation table comprising of four values $v_{b,c}$ where $v_{b,c}$ allows us to perform the transformation:

$$L_{u,b}, L_{v,c} \mapsto L_{w,G(b,c)}$$

The garbled circuits construction guarantees that given the translation table and labels L_{u,b^*} and L_{v,c^*} for specific input bits b^* and c^* , we can obtain $L_{w,G(b^*,c^*)}$; however, the other label at the output, namely $L_{w,1-G(b^*,c^*)}$ remains hidden.

In our setting, we replace labels with public keys, so that each wire is associated with a pair of public keys. As before, we also provide a translation table comprising four values $\text{rk}_{b,c}$ where the recoding key $\text{rk}_{b,c}$ allows us to perform the transformation

$$\text{Encode}(\text{pk}_{u,b}, s), \text{Encode}(\text{pk}_{v,c}, s) \mapsto \text{Encode}(\text{pk}_{w,G(b,c)}, s)$$

The security properties of the TOR scheme then give us the following guarantee: Given the translation table and encodings of s corresponding to b^*, c^* , we clearly compute the encoding

of s corresponding to $G(b^*, c^*)$. However, the encoding corresponding to $1 - G(b^*, c^*)$ remains pseudorandom.

Moreover, crucially, the translation table is independent of s , so we can now “reuse” the translation table by providing fresh encodings with different choices of s . In a sentence, replacing strings by functions gives us the power of reusability.

In the garbled circuits construction, the four entries of the table are permuted and thus, one can perform the translation even without knowing what the input bits b^* and c^* are. This is possible because there is an efficient way to verify when the “correct” translation key is being used. In contrast, in the reusable construction above, one has to know exactly which of the recoding keys to use. This is part of the reason why we are *unable to provide circuit or input privacy, but instead, only guarantee authenticity*, namely that an adversary can obtain only one of the two possible encodings at the output wire.

This construction forms the cornerstone of the subsequent work of Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich [GKP⁺13] who construct reusable garbled circuits with input and circuit privacy, by additionally leveraging the power of fully homomorphic encryption [Gen09, BV11].

From TOR to Attribute-Based Encryption. How is all this related to attribute-based encryption? In our attribute-based encryption scheme for circuits, the encodings of s are provided in the ciphertext, and the translation tables are provided in the secret key. More precisely, each wire is associated with two TOR public keys, and the encryption of a message m under an index ind is obtained by computing $\text{Encode}(\text{pk}_{i, \text{ind}_i}, s)$ for every input wire i . The output encoding $\text{Encode}(\text{pk}_{\text{out}}, s)$ is then used to mask the message. We obtain the secret key corresponding to a circuit C by “stitching” multiple translation tables together, where the public keys for the input and output wires are provided in the public parameters, and we pick fresh public keys for the internal wires during key generation. In a nutshell, this gives us the guarantee that given a secret key sk_C and an encryption $\text{Enc}(\text{ind}, m)$ such that $C(\text{ind}) = 1$, we can compute $\text{Encode}(\text{pk}_{\text{out}}, s)$ and thus recover the message. On the other hand, this value looks pseudorandom if $C(\text{ind}) = 0$.

In our outline of reusable garbled circuits with authenticity, we wanted to reuse the garbled circuit $G(C)$ across multiple encryptions with indices $\text{ind}_1, \text{ind}_2, \dots$ on which C always evaluates to 0. In attribute-based encryption, we also want reusability across multiple circuits C_1, C_2, \dots all of which evaluate to 0 on a fixed index ind (in addition to multiple indices). Fortunately, the strong security properties of the TOR primitive provide us with this guarantee.

To obtain attribute-based encryption for branching programs, we are able to support a different notion of translation tables, which we can realize using a slightly weaker notion of TOR. In branching programs, the transition function depends on an input variable and the current state. The fact that one of these two values is always an input variable makes things simpler; in circuits, both of the input values to a gate could be internal wires.

TOR from LWE. We show how to instantiate TOR from LWE, building upon previous lattice-based IBE techniques in [GPV08, CHKP12, ABB10a, ABB10b]. The public key is given by a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and

$$\text{Encode}(\mathbf{A}, s) = \mathbf{A}^T \mathbf{s} + \mathbf{e}$$

where $\mathbf{s} \in \mathbb{Z}_q^n$, $\mathbf{e} \in \mathbb{Z}_q^m$ is an error vector, and \mathbf{A}^T denotes the transpose of the matrix \mathbf{A} . (Correlated) pseudorandomness follows directly from the LWE assumption. Given $\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_{\text{tgt}} \in \mathbb{Z}_q^{n \times m}$, the recoding key \mathbf{rk} is given by a low-norm matrix $\mathbf{R} \in \mathbb{Z}_q^{2m \times m}$ such that

$$[\mathbf{A}_0 \parallel \mathbf{A}_1] \mathbf{R} = \mathbf{A}_{\text{tgt}}$$

Note that

$$\mathbf{R}^T \begin{bmatrix} \mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0 \\ \mathbf{A}_1^T \mathbf{s} + \mathbf{e}_1 \end{bmatrix} \approx \mathbf{A}_{\text{tgt}}^T \mathbf{s}$$

which gives us the recoding mechanism. There are three ways of generating the public key \mathbf{A}_{tgt} together with the recoding key \mathbf{R} : (1) using the trapdoor for \mathbf{A}_0 , (2) using the trapdoor for \mathbf{A}_1 , or (3) first generating \mathbf{R} and then “programming” $\mathbf{A}_{\text{tgt}} := [\mathbf{A}_0 \parallel \mathbf{A}_1] \mathbf{R}$. These three ways are statistically indistinguishable by the “bonsai trick” of [CHKP12]. In fact, our recoding mechanism is very similar to the lattice delegation mechanism introduced in [ABB10b], which also uses random low norm matrices to move from one lattice to another.

The multiplicative mechanism for recoding means that the noise grows exponentially with the number of sequential recodings. This, in turn, limits the depth of the circuits we can handle. In particular, the noise grows by a multiplicative $\text{poly}(n)$ factor on each recoding, which means that after depth d , it becomes $n^{O(d)}$. Since $n^{O(d)} < q/4 < 2^{n^\epsilon}$, we can handle circuits of depth $\tilde{O}(n^\epsilon)$ (here, the first inequality is for correctness and the second for security). Viewed differently, setting the LWE dimension $n = d^{1/\epsilon}$ lets us handle circuits of maximum depth $d = d(\ell)$.

Our weak TOR for branching programs uses an additive mechanism, namely the recoding key is given by a low-norm matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{A}_0 \mathbf{R} = \mathbf{A}_{\text{tgt}} - \mathbf{A}_1$. Note that $\mathbf{R}^T (\mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0) + (\mathbf{A}_1^T \mathbf{s} + \mathbf{e}_1) \approx \mathbf{A}_{\text{tgt}}^T \mathbf{s}$ which gives us our recoding mechanism. Since in our branching program construction, $\mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0$ will always be a fresh encoding provided in the ciphertext, the noise accumulation is additive rather than multiplicative.

2.3 Applications

Let us now explain the application of our result to the problem of publicly verifiable delegation of computation without input privacy.

A verifiable delegation scheme allows a computationally weak client to delegate expensive computations to the cloud, with the assurance that a malicious cloud cannot convince the client to accept an incorrect computation [Mic00, GKR08, GGP10, CKV10, AIK10]. Recent work of Parno, Raykova and Vaikuntanathan [PRV12] showed that any attribute-based encryption scheme for a class of circuits with encryption time at most linear in the length of the index immediately yields a two-message delegation scheme for the class in the pre-processing model. Namely, there is an initial pre-processing phase which fixes the circuit C the client wishes to compute, produces a circuit key and sends it to the server. Afterwards, to delegate computation on an input x , the client only needs to send a single message. Moreover, the ensuing delegation scheme satisfies public delegatability, namely anyone can delegate computations to the cloud; as well as public verifiability, namely anyone can check the cloud’s work (given a “verification” key published by the client). The previous delegation schemes that satisfy both these properties (secure in the standard model) supported the class NC^1 [PRV12, GPSW06, LW12]. Our attribute-based encryption schemes for circuits gives us a verifiable delegation scheme for all circuits, where the computation time of the client in the online phase is polynomial in the length of its input and the depth of the circuit, but

is otherwise independent of the circuit size. We note that this scheme does not guarantee privacy of the input. Building on this work, Goldwasser et al. [GKP⁺13] show how to achieve a publicly verifiable delegation scheme with input privacy.

2.4 Related Work

Prior to this work, the state-of-art for lattice-based predicate encryption was threshold and inner product predicates [ABV⁺12, AFV11]; realizing Boolean formula was itself an open problem. A different line of work considers definitional issues in the more general realm of functional encryption [BSW11, O’N10], for which general feasibility results are known for the restricted setting of a-priori bounded collusions developed from classical “one-time” garbled circuits [SS10a, GVW12] (the ciphertext size grows with both the circuit size and the collusion bound). Our methodology takes a fresh perspective on how to achieve reusability of garbled circuits with respect to authenticity. Our primitive (TOR) can be thought of as a generalization of the notion of proxy re-encryption [BBS98, AFGH06, HRSV11] which can be thought of as a one-to-one re-encryption mechanism.

Independent work. Boyen [Boy13] gave a construction of an ABE scheme for Boolean formulas based on LWE; our result for LWE-based branching program subsumes the result since Boolean formulas are a subclass of branching programs. Garg, Gentry, Halevi, Sahai and Waters [GGH⁺13] gave a construction of attribute-based encryption for general circuits under a DBDH-like assumption in multi-linear groups (unfortunately, there is no known candidate for realizing such an assumption), as well as a non-standard assumption in ideal lattices [GGH12]. The public parameters in the construction also grow with the depth of the circuit.

Subsequent Work. Our attribute-based encryption scheme has been used as the crucial component in the subsequent work of [GKP⁺13] to construct a (private index) functional encryption scheme with succinct ciphertexts. They also show a number of applications of their construction, including reusable garbled circuits *with input and circuit privacy*.

Organization. We present our TOR framework and its instantiation in Sections 4 and 5. We present our ABE scheme in Section 6. We present the scheme for branching programs in Section 7.

3 Preliminaries

Notation. Let PPT denote probabilistic polynomial-time. For any integer $q \geq 2$, we let \mathbb{Z}_q denote the ring of integers modulo q and we represent \mathbb{Z}_q as integers in $(-q/2, q/2]$. We let $\mathbb{Z}_q^{n \times m}$ denote the set of $n \times m$ matrices with entries in \mathbb{Z}_q . We use bold capital letters (e.g. \mathbf{A}) to denote matrices, bold lowercase letters (e.g. \mathbf{x}) to denote vectors. The notation \mathbf{A}^\top denotes the transpose of the matrix \mathbf{A} .

If \mathbf{A}_1 is an $n \times m$ matrix and \mathbf{A}_2 is an $n \times m'$ matrix, then $[\mathbf{A}_1 \parallel \mathbf{A}_2]$ denotes the $n \times (m + m')$ matrix formed by concatenating \mathbf{A}_1 and \mathbf{A}_2 . A similar notation applies to vectors. When doing matrix-vector multiplication we always view vectors as column vectors.

We say a function $f(n)$ is *negligible* if it is $O(n^{-c})$ for all $c > 0$, and we use $\text{negl}(n)$ to denote a negligible function of n . We say $f(n)$ is *polynomial* if it is $O(n^c)$ for some $c > 0$, and we use $\text{poly}(n)$ to denote a polynomial function of n . We say an event occurs with *overwhelming probability* if its

probability is $1 - \text{negl}(n)$. The function $\lg x$ is the base 2 logarithm of x . The notation $\lfloor x \rfloor$ denotes the nearest integer to x , rounding towards 0 for half-integers.

3.1 Attribute-Based Encryption

We define attribute-based encryption (ABE), following [GPSW06]. An ABE scheme for a class of predicate circuits \mathcal{C} (namely, circuits with a single bit output) consists of four algorithms (Setup, Enc, KeyGen, Dec):

$\text{Setup}(1^\lambda, 1^\ell) \rightarrow (\text{pp}, \text{mpk}, \text{msk})$: The setup algorithm gets as input the security parameter λ , the length ℓ of the index, and outputs the public parameter (pp, mpk) , and the master key msk . All the other algorithms get pp as part of its input.

$\text{Enc}(\text{mpk}, \text{ind}, m) \rightarrow \text{ct}_{\text{ind}}$: The encryption algorithm gets as input mpk , an index $\text{ind} \in \{0, 1\}^\ell$ and a message $m \in \mathcal{M}$. It outputs a ciphertext ct_{ind} . Note that ind is public given ct_{ind} .

$\text{KeyGen}(\text{msk}, C) \rightarrow \text{sk}_C$: The key generation algorithm gets as input msk and a predicate specified by $C \in \mathcal{C}$. It outputs a secret key sk_C (where C is also public).

$\text{Dec}(\text{sk}_C, \text{ct}_{\text{ind}}) \rightarrow m$: The decryption algorithm gets as input sk_C and ct_{ind} , and outputs either \perp or a message $m \in \mathcal{M}$.

We require that for all (ind, C) such that $C(\text{ind}) = 1$, all $m \in \mathcal{M}$ and $\text{ct}_{\text{ind}} \leftarrow \text{Enc}(\text{mpk}, \text{ind}, m)$, $\text{Dec}(\text{sk}_C, \text{ct}_{\text{ind}}) = m$.

Security Definition. For a stateful adversary \mathcal{A} , we define the advantage function $\text{Adv}_{\mathcal{A}}^{\text{PE}}(\lambda)$ to be

$$\Pr \left[b = b' : \begin{array}{l} \text{ind} \leftarrow \mathcal{A}(1^\lambda, 1^\ell); \\ (\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, 1^\ell); \\ (m_0, m_1) \leftarrow \mathcal{A}^{\text{KeyGen}(\text{msk}, \cdot)}(\text{mpk}), |m_0| = |m_1|; \\ b \xleftarrow{\$} \{0, 1\}; \\ \text{ct}_{\text{ind}} \leftarrow \text{Enc}(\text{mpk}, \text{ind}, m_b); \\ b' \leftarrow \mathcal{A}^{\text{KeyGen}(\text{msk}, \cdot)}(\text{ct}_{\text{ind}}) \end{array} \right] - \frac{1}{2}$$

with the restriction that all queries C that \mathcal{A} makes to $\text{KeyGen}(\text{msk}, \cdot)$ satisfies $C(\text{ind}) = 0$ (that is, sk_C does not decrypt ct_{ind}). an attribute-based encryption scheme is *selectively secure* if for all PPT adversaries \mathcal{A} , the advantage $\text{Adv}_{\mathcal{A}}^{\text{PE}}(\lambda)$ is a negligible function in λ . We call an attribute-based encryption scheme *fully secure* if the adversary \mathcal{A} is allowed to choose the challenge index ind after seeing secret keys, namely, along with choosing (m_0, m_1) .

3.2 Learning With Errors (LWE) Assumption

The LWE problem was introduced by Regev [Reg09], who showed that solving it *on the average* is as hard as (quantumly) solving several standard lattice problems *in the worst case*.

Definition 3.1 (LWE). For an integer $q = q(n) \geq 2$ and an error distribution $\chi = \chi(n)$ over \mathbb{Z}_q , the learning with errors problem $\text{dLWE}_{n,m,q,\chi}$ is to distinguish between the following pairs of distributions:

$$\{\mathbf{A}, \mathbf{A}\mathbf{s} + \mathbf{x}\} \quad \text{and} \quad \{\mathbf{A}, \mathbf{u}\}$$

where $\mathbf{A} \xleftarrow{\$} \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^n$, $\mathbf{x} \xleftarrow{\$} \chi^m$, $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_q^m$.

Connection to lattices. Let $B = B(n) \in \mathbb{N}$. A family of distributions $\chi = \{\chi_n\}_{n \in \mathbb{N}}$ is called B -bounded if

$$\Pr[\chi \in \{-B, \dots, B-1, B\}] = 1.$$

There are known quantum [Reg09] and classical [Pei09] reductions between $\text{dLWE}_{n,m,q,\chi}$ and approximating short vector problems in lattices in the worst case, where χ is a B -bounded (truncated) discretized Gaussian for some appropriate B . The state-of-the-art algorithms for these lattice problems run in time nearly exponential in the dimension n [AKS01, MV10]; more generally, we can get a 2^k -approximation in time $2^{\tilde{O}(n/k)}$. Combined with the connection to LWE, this means that the $\text{dLWE}_{n,m,q,\chi}$ assumption is quite plausible for a $\text{poly}(n)$ -bounded distribution χ and q as large as 2^{n^ϵ} (for any constant $0 < \epsilon < 1$). Throughout this paper, the parameter $m = \text{poly}(n)$, in which case we will shorten the notation slightly to $\text{LWE}_{n,q,\chi}$.

3.3 Trapdoors for Lattices and LWE

Gaussian distributions. Let $D_{\mathbb{Z}^m, \sigma}$ be the truncated discrete Gaussian distribution over \mathbb{Z}^m with parameter σ , that is, we replace the output by $\mathbf{0}$ whenever the $\|\cdot\|_\infty$ norm exceeds $\sqrt{m} \cdot \sigma$. Note that $D_{\mathbb{Z}^m, \sigma}$ is $\sqrt{m} \cdot \sigma$ -bounded.

Lemma 3.1 (Lattice Trapdoors [Ajt99, GPV08, MP12]). *There is an efficient randomized algorithm $\text{TrapSamp}(1^n, 1^m, q)$ that, given any integers $n \geq 1$, $q \geq 2$, and sufficiently large $m = \Omega(n \log q)$, outputs a parity check matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and a ‘trapdoor’ matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$ such that the distribution of \mathbf{A} is $\text{negl}(n)$ -close to uniform.*

Moreover, there is an efficient algorithm SampleD that with overwhelming probability over all random choices, does the following: For any $\mathbf{u} \in \mathbb{Z}_q^n$, and large enough $s = \Omega(\sqrt{n} \log q)$, the randomized algorithm $\text{SampleD}(\mathbf{A}, \mathbf{T}, \mathbf{u}, s)$ outputs a vector $\mathbf{r} \in \mathbb{Z}^m$ with norm $\|\mathbf{r}\|_\infty \leq \|\mathbf{r}\|_2 \leq s\sqrt{n}$ (with probability 1). Furthermore, the following distributions of the tuple $(\mathbf{A}, \mathbf{T}, \mathbf{U}, \mathbf{R})$ are within $\text{negl}(n)$ statistical distance of each other for any polynomial $k \in \mathbb{N}$:

- $(\mathbf{A}, \mathbf{T}) \leftarrow \text{TrapSamp}(1^n, 1^m, q)$; $\mathbf{U} \leftarrow \mathbb{Z}_q^{n \times k}$; $\mathbf{R} \leftarrow \text{SampleD}(\mathbf{A}, \mathbf{T}, \mathbf{U}, s)$.
- $(\mathbf{A}, \mathbf{T}) \leftarrow \text{TrapSamp}(1^n, 1^m, q)$; $\mathbf{R} \leftarrow (D_{\mathbb{Z}^m, s})^k$; $\mathbf{U} := \mathbf{A}\mathbf{R} \pmod{q}$.

4 Two-to-One Recoding Schemes

An overview is provided in Section 2.2.

Symmetric encryption. In our construction, we will use $\text{Encode}(\text{pk}, s)$ as a one-time key for a symmetric-key encryption scheme (E, D) . If Encode is deterministic, then we could simply use a one-time pad. However, since Encode is probabilistic, the one-time pad will not guarantee correctness. Instead, we require (E, D) to satisfy a stronger correctness guarantee, namely for all messages m and for all ψ, ψ' in the support $\text{Encode}(\text{pk}, s)$, $D(\psi', E(\psi, m)) = m$.

Allowing degradation. With each recoding operation, the “quality” of encoding potentially degrades. In order to formalize this, we allow the initial global public parameters to depend on d_{\max} , an a-prior upper bound on the number of nested recoding operations. We then require that given any encodings ψ and ψ' that are a result of at most d_{\max} nested recodings, $D(\psi', E(\psi, m)) = m$. We stress that we allow d_{\max} to be super-polynomial, and in fact, provide such instantiations for a relaxed notion of TOR.

4.1 Definition of TOR

Formally, a TOR scheme over the input space $\mathcal{S} = \{\mathcal{S}_\lambda\}$ consists of six polynomial-time algorithms (Params , Keygen , Encode , ReKeyGen , SimReKeyGen , Recode) and a symmetric-key encryption scheme (E, D) with the following properties:

- $\text{Params}(1^\lambda, d_{\max})$ is a probabilistic algorithm that takes as input the security parameter λ and an upper bound d_{\max} on the number of nested recoding operations (written in binary), outputs “global” public parameters pp .
- $\text{Keygen}(\text{pp})$ is a probabilistic algorithm that outputs a public/secret key pair (pk, sk) .
- $\text{Encode}(\text{pk}, s)$ is a probabilistic algorithm that takes pk and an input $s \in \mathcal{S}$, and outputs an encoding ψ .

In addition, there is a recoding mechanism together with two ways to generate recoding keys: given one of the two secret keys, or by programming the output public key.

- $\text{ReKeyGen}(\text{pk}_0, \text{pk}_1, \text{sk}_0, \text{pk}_{\text{tgt}})$ is a probabilistic algorithm that takes a key pair $(\text{pk}_0, \text{sk}_0)$, another public key pk_1 , a “target” public key pk_{tgt} , and outputs a recoding key rk .
- $\text{SimReKeyGen}(\text{pk}_0, \text{pk}_1)$ is a probabilistic algorithm that takes two public keys pk_0, pk_1 and outputs a recoding key rk together with a “target” public key pk_{tgt} .
- $\text{Recode}(\text{rk}, \psi_0, \psi_1)$ is a deterministic algorithm that takes the recoding key rk , two encodings ψ_0 and ψ_1 , and outputs an encoding ψ_{tgt} .

Remark 4.1. For our instantiation from lattices, we can in fact invert $\text{Encode}(\text{pk}, s)$ to recover s using the corresponding sk . However, we will not require this property in our generic constructions from TOR. Indeed, realizing this property over bilinear groups would be hard, since s is typically encoded in the exponent.

Correctness. Correctness of a TOR scheme requires two things. First, for every pk and $s \in \mathcal{S}$, there exists a family of sets $\Psi_{\text{pk},s,j}, j = 0, 1, \dots, d_{\max}$:

- $\Pr[\text{Encode}(\text{pk}, s) \in \Psi_{\text{pk},s,0}] = 1$, where the probability is taken over the coin tosses of Encode ;
- $\Psi_{\text{pk},s,0} \subseteq \Psi_{\text{pk},s,1} \subseteq \dots \subseteq \Psi_{\text{pk},s,d_{\max}}$.
- for all $\psi, \psi' \in \Psi_{\text{pk},s,d_{\max}}$ and all $m \in \mathcal{M}$, $D(\psi', E(\psi, m)) = m$.

Note that these properties hold trivially if Encode is deterministic and (E, D) is the one-time pad. Secondly, the correctness of recoding requires that for any triple of key pairs $(\text{pk}_0, \text{sk}_0), (\text{pk}_1, \text{sk}_1), (\text{pk}_{\text{tgt}}, \text{sk}_{\text{tgt}})$, and any encodings $\psi_0 \in \Psi_{\text{pk}_0,s,j_0}$ and $\psi_1 \in \Psi_{\text{pk}_1,s,j_1}$,

$$\text{Recode}(\text{rk}, \psi_0, \psi_1) \in \Psi_{\text{pk}_{\text{tgt}},s,\max(j_0,j_1)+1}$$

Statistical Security Properties. Note that we have three ways of sampling recoding keys: using ReKeyGen along with one of two secret keys sk_0 or sk_1 ; using SimReKeyGen while programming pk_{tgt} . We require that all three ways lead to the same distribution of recoding keys, up to some statistical error.

(Key Indistinguishability) : Let $(\text{pk}_b, \text{sk}_b) \leftarrow \text{Keygen}(\text{pp})$ for $b = 0, 1$ and $(\text{pk}_{\text{tgt}}, \text{sk}_{\text{tgt}}) \leftarrow \text{Keygen}(\text{pp})$.

The following two ensembles must be statistically indistinguishable:

$$\left[\text{Aux}, \text{ReKeyGen}(\text{pk}_0, \text{pk}_1, \boxed{\text{sk}_0}, \text{pk}_{\text{tgt}}) \right] \stackrel{s}{\approx} \left[\text{Aux}, \text{ReKeyGen}(\text{pk}_1, \text{pk}_0, \boxed{\text{sk}_1}, \text{pk}_{\text{tgt}}) \right]$$

where $\text{Aux} = ((\text{pk}_0, \text{sk}_0), (\text{pk}_1, \text{sk}_1), (\text{pk}_{\text{tgt}}, \text{sk}_{\text{tgt}}))$. Informally, this says that sampling recoding keys using sk_0 or sk_1 yields the same distribution.

(Recoding Simulation) : Let $(\text{pk}_b, \text{sk}_b) \leftarrow \text{Keygen}(\text{pp})$ for $b = 0, 1$. Then, the following two ways of sampling the tuple $[(\text{pk}_0, \text{sk}_0), (\text{pk}_1, \text{sk}_1), \text{pk}_{\text{tgt}}, \text{rk}]$ must be statistically indistinguishable:

$$\left[(\text{pk}_0, \text{sk}_0), (\text{pk}_1, \text{sk}_1), \text{pk}_{\text{tgt}}, \text{rk} : (\text{pk}_{\text{tgt}}, \text{sk}_{\text{tgt}}) \leftarrow \text{Keygen}(\text{pp}); \text{rk} \leftarrow \text{ReKeyGen}(\text{pk}_0, \text{pk}_1, \text{sk}_0, \text{pk}_{\text{tgt}}) \right] \stackrel{s}{\approx} \left[(\text{pk}_0, \text{sk}_0), (\text{pk}_1, \text{sk}_1), \text{pk}_{\text{tgt}}, \text{rk} : (\text{pk}_{\text{tgt}}, \text{rk}) \leftarrow \text{SimReKeyGen}(\text{pk}_0, \text{pk}_1) \right]$$

In addition, we require one-time semantic security for (E, D) :

(One-time Semantic Security) : For all $m_0, m_1 \in \mathcal{M}$, the following two distributions must be statistically indistinguishable:

$$\left[E(\psi, m_0) : \psi \xleftarrow{\$} \mathcal{K} \right] \stackrel{s}{\approx} \left[E(\psi, m_1) : \psi \xleftarrow{\$} \mathcal{K} \right]$$

For all three properties, computational indistinguishability is sufficient for our applications, but we will achieve the stronger statistical indistinguishability in our instantiations.

Computational Security Property. We require that given the encoding of a random s on $\ell = \text{poly}(\lambda)$ keys, the evaluation at a fresh key is pseudorandom.

(Correlated Pseudorandomness) : For every polynomial $\ell = \ell(\lambda)$, let $(\text{pk}_i, \text{sk}_i) \leftarrow \text{Keygen}(\text{pp})$ for $i \in [\ell + 1]$. Let $s \xleftarrow{\$} \mathcal{S}$, and let $\psi_i \leftarrow \text{Encode}(\text{pk}_i, s)$ for $i \in [\ell + 1]$. Then, the following two ensembles must be computationally indistinguishable:

$$\left[(\text{pk}_i, \psi_i)_{i \in [\ell]}, \text{pk}_{\ell+1}, \boxed{\psi_{\ell+1}} \right] \stackrel{c}{\approx} \left[(\text{pk}_i, \psi_i)_{i \in [\ell]}, \text{pk}_{\ell+1}, \boxed{\psi} : \psi \xleftarrow{\$} \mathcal{K} \right]$$

That is, we define the advantage function $\text{Adv}_{\mathcal{A}}^{\text{CP}}(\lambda)$ to be:

$$\Pr \left[\begin{array}{l} \text{pp} \leftarrow \text{Setup}(1^\lambda); s \leftarrow \mathcal{S}; \\ (\text{pk}_i, \text{sk}_i) \leftarrow \text{Keygen}(\text{pp}), \\ \psi_i \leftarrow \text{Encode}(\text{pk}_i, s), i = 1, \dots, \ell; \\ \psi'_0 \leftarrow \text{Encode}(\text{pk}_{\ell+1}, s); \\ b \xleftarrow{\$} \{0, 1\}; \psi'_1 \xleftarrow{\$} \mathcal{K} \\ b' \leftarrow \mathcal{A}(\text{pk}_1, \dots, \text{pk}_{\ell+1}, \psi_1, \dots, \psi_\ell, \psi'_b) \end{array} \right] - \frac{1}{2}$$

and we require that for all PPT \mathcal{A} , the advantage function $\text{Adv}_{\mathcal{A}}^{\text{CP}}(\lambda)$ is a negligible function in λ .

4.2 Simple Applications of TOR

First example. We revisit the example from Section 2.2. Consider a two-input boolean gate g with input wires u, v and output wire w , computing a function $G : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$. Analogous to Yao's garbled circuit, we provide a translation table Γ comprising four values

$$\Gamma := (\text{rk}_{b,c} : b, c \in \{0, 1\})$$

where $\text{rk}_{b,c}$ allows us to perform the transformation

$$\text{Encode}(\text{pk}_{u,b}, s), \text{Encode}(\text{pk}_{v,c}, s) \mapsto \text{Encode}(\text{pk}_{w,G(b,c)}, s)$$

Now, fix b^*, c^* and $d^* := G(b^*, c^*)$. Given an encoding of s corresponding to b^* and c^* , we can compute that under for d^* using the recoding key rk_{b^*, c^*} ; in addition, we claim that the encoding corresponding to $1 - d^*$ remains pseudorandom. To prove this, it suffices to simulate Γ given $\text{pk}_{u, b^*}, \text{pk}_{v, c^*}, \text{pk}_{w, 1-d^*}$ as follows:

- we sample $(\text{pk}_{w, d^*}, \text{rk}_{b^*, c^*})$ using SimReKeyGen ;
- we sample $\text{pk}_{u, 1-b^*}$ and $\text{pk}_{v, 1-c^*}$ along with the corresponding secret keys; using these secret keys, we can sample the other three recoding keys $\text{rk}_{1-b^*, c^*}, \text{rk}_{b^*, 1-c^*}, \text{rk}_{1-b^*, 1-c^*}$.

IBE from TOR. As a warm-up, we show how to build a selectively secure IBE for identity space $\{0, 1\}^\ell$.

$$\text{mpk} := \begin{pmatrix} \text{pk}_{1,0} & \text{pk}_{2,0} & \dots & \text{pk}_{\ell,0} & \text{pk}_{\text{start}} \\ \text{pk}_{1,1} & \text{pk}_{2,1} & \dots & \text{pk}_{\ell,1} & \text{pk}_{\text{out}} \end{pmatrix}$$

The ciphertext for identity ind and message m is given by:

$$\left(\text{Encode}(\text{pk}_{1,\text{ind}_1}, s), \dots, \text{Encode}(\text{pk}_{\ell,\text{ind}_\ell}, s), \text{Encode}(\text{pk}_{\text{start}}, s), \text{E}(\text{Encode}(\text{pk}_{\text{out}}, s), m) \right)$$

The secret key for identity ind is given by $(\text{rk}_1, \dots, \text{rk}_\ell)$ where we first sample

$$(\text{pk}'_1, \text{sk}'_1), \dots, (\text{pk}'_{\ell-1}, \text{sk}'_{\ell-1}) \leftarrow \text{Keygen}(\text{pp})$$

and then sample

$$\begin{aligned} \text{rk}_1 &\leftarrow \text{ReKeyGen}(\text{pk}_{\text{start}}, \text{pk}_{1,\text{ind}_1}, \text{sk}_{\text{start}}, \text{pk}'_1) \\ \text{rk}_2 &\leftarrow \text{ReKeyGen}(\text{pk}'_1, \text{pk}_{2,\text{ind}_2}, \text{sk}'_1, \text{pk}'_2) \\ &\vdots \\ \text{rk}_\ell &\leftarrow \text{ReKeyGen}(\text{pk}'_{\ell-1}, \text{pk}_{\ell,\text{ind}_\ell}, \text{sk}'_{\ell-1}, \text{pk}_{\text{out}}) \end{aligned}$$

To prove selective security, we need to generate secret keys for any $\text{ind} \neq \text{ind}^*$, given $\text{sk}_{1,1-\text{ind}_1^*}, \dots, \text{sk}_{\ell,1-\text{ind}_\ell^*}$ but not sk_{start} or sk_{out} . We can achieve this as follows: pick an i for which $\text{ind}_i \neq \text{ind}_i^*$;

- pick $(\text{rk}_1, \text{pk}'_1), \dots, (\text{rk}_{i-1}, \text{pk}'_{i-1})$ using SimReKeyGen ;
- pick $(\text{pk}'_i, \text{sk}'_i), \dots, (\text{pk}'_{\ell-1}, \text{sk}'_{\ell-1})$ using Keygen ;
- pick $\text{rk}_i, \text{rk}_{i+1}, \dots, \text{rk}_\ell$ using ReKeyGen with secret keys $\text{sk}_{1-\text{ind}_i^*}, \text{sk}'_i, \dots, \text{sk}'_{\ell-1}$ respectively.

5 TOR from LWE

In this section, we present an instantiation of TOR from LWE, building upon ideas previously introduced in [GPV08, CHKP12, ABB10a, ABB10b].

Lemma 5.1. *Assuming $\text{dLWE}_{n,q,\chi}$ where $q = n^{\Theta(d_{\max})}$, there is a TOR scheme that is correct up to d_{\max} levels.*

- **Params**($1^\lambda, d_{\max}$): First choose the LWE dimension $n = n(\lambda)$. Let the error distribution $\chi = \chi(n) = D_{\mathbb{Z}, \sqrt{n}}$, the error bound $B = B(n) = O(n)$, the modulus $q = q(n) = \tilde{O}(n^2 d_{\max}^{d_{\max}} n)$, the number of samples $m = m(n) = O(n \log q)$ and the Gaussian parameter $s = s(n) = O(\sqrt{n \log q})$. Output the global public parameters $\text{pp} = (n, \chi, B, q, m, s)$. Define the domain \mathcal{S} of the encoding scheme to be \mathbb{Z}_q^n .
- **Keygen**(pp): Run the trapdoor generation algorithm $\text{TrapGen}(1^n, 1^m, q)$ to obtain a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with the trapdoor matrix $\mathbf{T} \in \mathbb{Z}^{m \times m}$. Output $\text{pk} := \mathbf{A}$ and $\text{sk} := \mathbf{T}$.
- **Encode**(pk, s): Sample an error vector $\mathbf{e} \leftarrow \chi^m$ and output the encoding $\psi := \mathbf{A}^T \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^n$.

The recoding algorithms work as follows:

- **ReKeyGen**($\mathbf{pk}_0, \mathbf{pk}_1, \mathbf{sk}_b, \mathbf{pk}_{\text{tgt}}$): Let $\mathbf{pk}_0 = \mathbf{A}_0$, $\mathbf{pk}_1 = \mathbf{A}_1$, $\mathbf{sk}_b = \mathbf{T}_b$ and $\mathbf{pk}_{\text{tgt}} = \mathbf{A}_{\text{tgt}}$. Compute the matrix $\mathbf{R} \in \mathbb{Z}^{2m \times m}$ in the following way:
 - Choose a discrete Gaussian matrix $\mathbf{R}_{1-b} \leftarrow (D_{\mathbb{Z},s})^{m \times m}$. Namely, each entry of the matrix is an independent sample from the discrete Gaussian distribution $D_{\mathbb{Z},s}$.
 - Compute $\mathbf{U} := \mathbf{A}_{\text{tgt}} - \mathbf{A}_{1-b}\mathbf{R}_{1-b} \in \mathbb{Z}_q^{n \times m}$.
 - Compute the matrix \mathbf{R}_b by running the algorithm **SampleD** to compute a matrix $\mathbf{R}_b \in \mathbb{Z}^{m \times m}$ as follows:

$$\mathbf{R}_b \leftarrow \text{SampleD}(\mathbf{A}_b, \mathbf{T}_b, \mathbf{U})$$

Output

$$\mathbf{rk}_{0,1}^{\text{tgt}} := \begin{bmatrix} \mathbf{R}_0 \\ \mathbf{R}_1 \end{bmatrix} \in \mathbb{Z}^{2m \times m}$$

(We remark that $\mathbf{A}_b\mathbf{R}_b = \mathbf{U} = \mathbf{A}_{\text{tgt}} - \mathbf{A}_{1-b}\mathbf{R}_{1-b}$, and thus, $\mathbf{A}_0\mathbf{R}_0 + \mathbf{A}_1\mathbf{R}_1 = \mathbf{A}_{\text{tgt}}$).

- **SimReKeyGen**($\mathbf{pk}_0, \mathbf{pk}_1$): Let $\mathbf{pk}_0 = \mathbf{A}_0$ and $\mathbf{pk}_1 = \mathbf{A}_1$.
 - Sample a matrix $\mathbf{R} \leftarrow (D_{\mathbb{Z},s})^{2m \times m}$ by sampling each entry from the discrete Gaussian distribution $D_{\mathbb{Z},s}$.
 - Define

$$\mathbf{A}_{\text{tgt}} := [\mathbf{A}_0 \parallel \mathbf{A}_1] \mathbf{R} \in \mathbb{Z}_q^{n \times m}$$

Output the pair $(\mathbf{pk}_{\text{tgt}} := \mathbf{A}_{\text{tgt}}, \mathbf{rk}_{0,1}^{\text{tgt}} := \mathbf{R})$.

- **Recode**($\mathbf{rk}_{0,1}^{\text{tgt}}, \psi_0, \psi_1$): Let $\mathbf{rk}_{0,1}^{\text{tgt}} = \mathbf{R}$. Compute the recoded ciphertext

$$\psi_{\text{tgt}} = [\psi_0^T \parallel \psi_1^T] \mathbf{R}$$

We also need a one-time symmetric encryption scheme (\mathbf{E}, \mathbf{D}) which we will instantiate as an *error-tolerant* version of the one-time pad with $\mathcal{K} = \mathbb{Z}_q^n, \mathcal{M} = \{0, 1\}^n$, as follows:

- $\mathbf{E}(\psi, \mu)$ takes as input a vector $\psi \in \mathbb{Z}_q^n$ and a bit string $\mu \in \mathcal{M}$ and outputs the encryption

$$\gamma := \psi + \lceil q/2 \rceil \mu \pmod{q}$$

- $\mathbf{D}(\psi', \gamma)$ takes as input a vector $\psi' = (\psi'_1, \dots, \psi'_n) \in \mathbb{Z}_q^n$, an encryption $\gamma = (\gamma_1, \dots, \gamma_n) \in \mathbb{Z}_q^n$ and does the following. Define a function $\text{Round}(x)$ where $x \in [-(q-1)/2, \dots, (q-1)/2]$ as:

$$\text{Round}(x) = \begin{cases} 0 & \text{if } |x| < q/4 \\ 1 & \text{otherwise} \end{cases}$$

The decryption algorithm outputs a vector $\mu = (\text{Round}(\gamma_1 - \psi'_1), \dots, \text{Round}(\gamma_n - \psi'_n))$.

We defer the analysis of (\mathbf{E}, \mathbf{D}) to the full version.

5.1 Analysis

Correctness. We define the sets $\Psi_{\mathbf{A},\mathbf{s},j}$ for $\mathbf{pk} := \mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $j \in [1 \dots d_{\max}]$ as follows:

$$\Psi_{\mathbf{A},\mathbf{s},j} = \{\mathbf{A}^T \mathbf{s} + \mathbf{e} : \|\mathbf{e}\|_\infty \leq B \cdot (2sm\sqrt{m})^j\}$$

Given this definition:

- Observe that when $\mathbf{e} \leftarrow \chi^m$, $\|\mathbf{e}\|_\infty \leq B$ by the definition of χ and B . $\Pr[\text{Encode}(\mathbf{A}, \mathbf{s}) \in \Psi_{\mathbf{A},\mathbf{s},0}] = 1$.
- $\Psi_{\mathbf{A},\mathbf{s},0} \subseteq \Psi_{\mathbf{A},\mathbf{s},1} \subseteq \dots \subseteq \Psi_{\mathbf{A},\mathbf{s},d_{\max}}$, by definition of the sets above.
- For any two encodings $\psi = \mathbf{A}^T \mathbf{s} + \mathbf{e}$, $\psi' = \mathbf{A}^T \mathbf{s} + \mathbf{e}' \in \Psi_{\mathbf{A},\mathbf{s},d_{\max}}$,

$$\|\psi - \psi'\|_\infty = \|\mathbf{e} - \mathbf{e}'\|_\infty \leq 2 \cdot B \cdot (2sm\sqrt{m})^{d_{\max}} < q/4,$$

which holds as long as $n \cdot O(n^2 \log q)^{d_{\max}} < q/4$. Thus, ψ and ψ' are “close”, and by the correctness property of the symmetric encryption scheme (\mathbf{E}, \mathbf{D}) described above, $\mathbf{D}(\psi', \mathbf{E}(\psi, \mu)) = \mu$ for any $\mu \in \{0, 1\}^n$.

- Consider two encodings $\psi_0 \in \Psi_{\mathbf{A}_0, \mathbf{s}, j_0}$ and $\psi_1 \in \Psi_{\mathbf{A}_1, \mathbf{s}, j_1}$ for any $j_0, j_1 \in \mathbb{N}$, any $\mathbf{A}_0, \mathbf{A}_1 \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{s} \in \mathbb{Z}_q^n$. Then, $\psi_0 = \mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0$ and $\psi_1 := \mathbf{A}_1^T \mathbf{s} + \mathbf{e}_1$ where $\|\mathbf{e}_0\|_\infty \leq B \cdot (2sm\sqrt{m})^{j_0}$ and $\|\mathbf{e}_1\|_\infty \leq B \cdot (2sm\sqrt{m})^{j_1}$.

Then, the recoded ciphertext ψ_{tgt} is computed as follows:

$$\begin{aligned} \psi_{\text{tgt}}^T &:= [\psi_0^T \parallel \psi_1^T] \mathbf{R}_{0,1}^{\text{tgt}} \\ &= [\mathbf{s}^T \mathbf{A}_0 + \mathbf{e}_0^T \parallel \mathbf{s}^T \mathbf{A}_1 + \mathbf{e}_1^T] \mathbf{R}_{0,1}^{\text{tgt}} \\ &= \mathbf{s}^T [\mathbf{A}_0 \parallel \mathbf{A}_1] \mathbf{R}_{0,1}^{\text{tgt}} + [\mathbf{e}_0^T \parallel \mathbf{e}_1^T] \mathbf{R}_{0,1}^{\text{tgt}} \\ &= \mathbf{s}^T \mathbf{A}_{\text{tgt}} + \mathbf{e}_{\text{tgt}} \end{aligned}$$

where the last equation is because $\mathbf{A}_{\text{tgt}} = [\mathbf{A}_0 \parallel \mathbf{A}_1] \mathbf{R}_{0,1}^{\text{tgt}}$ and we define $\mathbf{e}_{\text{tgt}} := [\mathbf{e}_0^T \parallel \mathbf{e}_1^T] \mathbf{R}_{0,1}^{\text{tgt}}$. Thus,

$$\begin{aligned} \|\mathbf{e}_{\text{tgt}}\|_\infty &\leq m \cdot \|\mathbf{R}_{0,1}^{\text{tgt}}\|_\infty \cdot (\|\mathbf{e}_0\|_\infty + \|\mathbf{e}_1\|_\infty) \\ &\leq m \cdot s\sqrt{m} \cdot (B \cdot (2sm\sqrt{m})^{j_0} + B \cdot (2sm\sqrt{m})^{j_1}) \\ &\leq B \cdot (2sm\sqrt{m})^{\max(j_0, j_1)+1} \end{aligned}$$

exactly as required. Here, the second inequality is because $\|\mathbf{R}_{0,1}^{\text{tgt}}\|_\infty \leq s\sqrt{m}$ by Lemma 3.1. This finishes our proof of correctness.

Key Indistinguishability. Recall that in ReKeyGen, we given sampling $(\mathbf{R}_0, \mathbf{R}_1)$ satisfying $\mathbf{A}_0 \mathbf{R}_0 + \mathbf{A}_1 \mathbf{R}_1 = \mathbf{A}_{\text{tgt}}$. Key indistinguishability basically says that we obtain the same distribution whether we use a trapdoor for \mathbf{A}_0 or that for \mathbf{A}_1 . Indeed, this follows directly from the following statement in [CHKP12, GPV08] (see also [CHK09, Theorem 3.4]): for every $(\mathbf{A}_0, \mathbf{T}_0)$, $(\mathbf{A}_1, \mathbf{T}_1)$ generated by $\text{TrapSamp}(1^n, 1^m, q)$, every matrix $\mathbf{V} \in \mathbb{Z}_q^{n \times m}$, and any $s = \Omega(\sqrt{n \log q})$, the following two experiments generate distributions with $\text{negl}(n)$ statistical distance:

- Sample $\mathbf{R}_0 \leftarrow (D_{\mathbb{Z}^m, s})^m$, compute $\mathbf{U} := \mathbf{V} - \mathbf{A}_0 \mathbf{R}_0 \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{R}_1 \leftarrow \text{SampleD}(\mathbf{A}_1, \mathbf{T}_1, \mathbf{U}, s)$. Output $(\mathbf{R}_0, \mathbf{R}_1)$.
- Sample $\mathbf{R}_1 \leftarrow (D_{\mathbb{Z}^m, s})^m$, compute $\mathbf{U} := \mathbf{V} - \mathbf{A}_1 \mathbf{R}_1 \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{R}_0 \leftarrow \text{SampleD}(\mathbf{A}_0, \mathbf{T}_0, \mathbf{U}, s)$. Output $(\mathbf{R}_0, \mathbf{R}_1)$.

The recoding simulation property follows readily from Lemma 3.1, as is done in [CHKP12]. Correlated pseudorandomness directly from the decisional LWE assumption $\text{dLWE}_{n, (\ell+1) \cdot m, q, \chi}$ where $q = n^{\Theta(d_{\max})}$.

6 Attribute-Based Encryption for Circuits

In this section, we show how to construct attribute-based encryption for circuits from any TOR scheme. Let TOR be the scheme consisting of algorithms (Params, Keygen, Encode) with the “two-to-one” recoding mechanism (Recode, ReKeyGen, SimReKeyGen) with input space \mathcal{S} . For every d_{\max} , let d_{\max} -TOR denote a secure “two-to-one” recoding scheme that is correct for d_{\max} recoding levels.

Theorem 6.1. *For every ℓ and polynomial $d_{\max} = d_{\max}(\lambda)$, let $\mathcal{C}_{\ell, d_{\max}}$ denote a family of polynomial-size circuits of depth at most d_{\max} that take ℓ bits of input. Assuming the existence of a d_{\max} -TOR scheme, there exists a selectively secure attribute-based encryption scheme \mathcal{ABE} for \mathcal{C} .*

Combining Theorem 6.1 and Lemma 5.1, we obtain a selectively secure attribute-based encryption scheme from LWE. Furthermore, invoking an argument from [BB04, Theorem 7.1] and using subexponential hardness of LWE, we obtain a fully secure scheme:

Corollary 6.2. *For all ℓ and polynomial $d_{\max} = d_{\max}(\ell)$, there exists a selectively secure attribute-based encryption scheme \mathcal{ABE} for any family of polynomial-size circuits with ℓ inputs and depth at most d_{\max} , assuming the hardness of $\text{dLWE}_{n, q, \chi}$ for sufficiently large $n = \text{poly}(\lambda, d_{\max})$, $q = n^{O(d_{\max})}$ and some $\text{poly}(n)$ -bounded error distribution χ .*

Moreover, assuming $2^{O(\ell)}$ -hardness of $\text{dLWE}_{n, q, \chi}$ for parameters $n = \text{poly}(\lambda, d_{\max}, \ell)$, and q and χ as above, the attribute-based encryption scheme \mathcal{ABE} is fully secure.

The reader is referred to the text after the construction for further explanation of how to choose the LWE parameters.

Observe that if we start with a TOR scheme that supports $d_{\max} = \ell^{\omega(1)}$, then our construction immediately yields an attribute-based encryption scheme for arbitrary polynomial-size circuit families (without any restriction on the depth). This can be achieved if, for example, we had an LWE-based TOR scheme where q grows polynomially instead of exponentially in d_{\max} as in our LWE-based weak TOR.

We now prove Theorem 6.1.

Circuit Representation. Let \mathcal{C}_λ be a collection of circuits each having $\ell = \ell(\lambda)$ input wires and one output wire. Define a collection $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$. For each $C \in \mathcal{C}_\lambda$, we index the wires of C in the following way. The input wires are indexed 1 to ℓ , the internal wires have indices $\ell + 1, \ell + 2, \dots, |C| - 1$ and the output wire has index $|C|$, which also denotes the size of the circuit. We assume that the circuit is composed of arbitrary two-to-one gates. Each gate g is indexed as

a tuple (u, v, w) where u and v are the incoming wire indices, and $w > \max\{u, v\}$ is the outgoing wire index. The gate computes the function $g_w : \{0, 1\} \times \{0, 1\} \rightarrow \{0, 1\}$. The “fan-out wires” in the circuit are given a single number. That is, if the outgoing wire of a gate feeds into the input of multiple gates, then all these wires are indexed the same. (See e.g. [BHR12, Fig 4].)

6.1 Construction from TOR

The ABE scheme $\mathcal{ABE} = (\text{Setup}, \text{Enc}, \text{KeyGen}, \text{Dec})$ is defined as follows.

$\text{Setup}(1^\lambda, 1^\ell, d_{\max})$: For each of the ℓ input wires, generate two public/secret key pairs. Also, generate an additional public/secret key pair:

$$\begin{aligned} (\text{pk}_{i,b}, \text{sk}_{i,b}) &\leftarrow \text{Keygen}(\text{pp}) \quad \text{for } i \in [\ell], b \in \{0, 1\} \\ (\text{pk}_{\text{out}}, \text{sk}_{\text{out}}) &\leftarrow \text{Keygen}(\text{pp}) \end{aligned}$$

Output

$$\text{mpk} := \begin{pmatrix} \text{pk}_{1,0} & \text{pk}_{2,0} & \dots & \text{pk}_{\ell,0} \\ \text{pk}_{1,1} & \text{pk}_{2,1} & \dots & \text{pk}_{\ell,1} & \text{pk}_{\text{out}} \end{pmatrix} \quad \text{msk} := \begin{pmatrix} \text{sk}_{1,0} & \text{sk}_{2,0} & \dots & \text{sk}_{\ell,0} \\ \text{sk}_{1,1} & \text{sk}_{2,1} & \dots & \text{sk}_{\ell,1} \end{pmatrix}$$

$\text{Enc}(\text{mpk}, \text{ind}, m)$: For $\text{ind} \in \{0, 1\}^\ell$, choose a uniformly random $s \xleftarrow{\$} \mathcal{S}$ and encode it under the public keys specified by the index bits:

$$\psi_i \leftarrow \text{Encode}(\text{pk}_{i, \text{ind}_i}, s) \text{ for all } i \in [\ell]$$

Encrypt the message m :

$$\tau \leftarrow \text{E}(\text{Encode}(\text{pk}_{\text{out}}, s), m)$$

Output the ciphertext

$$\text{ct}_{\text{ind}} := (\psi_1, \psi_2, \dots, \psi_\ell, \tau)$$

$\text{KeyGen}(\text{msk}, C)$:

1. For every non-input wire $w = \ell + 1, \dots, |C|$ of the circuit C , and every $b \in \{0, 1\}$, generate public/secret key pairs:

$$(\text{pk}_{w,b}, \text{sk}_{w,b}) \leftarrow \text{Keygen}(\text{pp}) \text{ if } w < |C| \text{ or } b = 0$$

and set $\text{pk}_{|C|,1} := \text{pk}_{\text{out}}$.

2. For the gate $g = (u, v, w)$ with outgoing wire w , compute the four recoding keys $\text{rk}_{b,c}^w$ (for $b, c \in \{0, 1\}$):

$$\text{rk}_{b,c}^w \leftarrow \text{ReKeyGen}(\text{pk}_{u,b}, \text{pk}_{v,c}, \text{sk}_{u,b}, \text{pk}_{w, g_w(b,c)})$$

Output the secret key which is a collection of $4(|C| - \ell)$ recoding keys

$$\text{sk}_C := (\text{rk}_{b,c}^w : w \in [\ell + 1, |C|], b, c \in \{0, 1\})$$

$\text{Dec}(\text{sk}_C, \text{ct}_{\text{ind}})$: We tacitly assume that ct_{ind} contains the index ind . For $w = \ell + 1, \dots, |C|$, let $g = (u, v, w)$ denote the gate with outgoing wire w . Suppose wires u and v carry the values b^* and c^* , so that wire w carries the value $d^* := g_w(b^*, c^*)$. Compute

$$\psi_{w,d^*} \leftarrow \text{Recode}\left(\text{rk}_{b^*,c^*}^w, \psi_{u,b^*}, \psi_{v,c^*}\right)$$

If $C(\text{ind}) = 1$, then we would have computed $\psi_{|C|,1}$. Output the message

$$m \leftarrow D(\psi_{|C|,1}, \tau)$$

If $C(\text{ind}) = 0$, output \perp .

LWE Parameters. Fix $\ell = \ell(\lambda)$ and $d_{\max} = d_{\max}(\ell)$, and suppose the $\text{dLWE}_{n,m,q,\chi}$ assumption holds for $q = 2^{n^\epsilon}$ for some $0 < \epsilon < 1$. Then, in our LWE-based TOR, we will set:

$$n = \tilde{\Theta}(d_{\max}^{1/\epsilon}) \quad \text{and} \quad q = n^{\Theta(d_{\max})}$$

By Corollary 6.2, we get security under 2^{n^ϵ} -LWE.

6.2 Correctness

Lemma 6.3 (correctness). *Let $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be family where each \mathcal{C}_λ is a finite collection of polynomial-size circuits each of depth at most d_{\max} . Let TOR be a correct “two-to-one” recoding scheme for d_{\max} levels. Then, the construction presented above is a correct attribute-based encryption scheme.*

Proof. Fix a circuit C of depth at most d_{\max} and an input ind such that $C(\text{ind}) = 1$. Informally, we rely on recoding correctness for d_{\max} recodings to show that $w = 1, \dots, |C|$, we have

$$\psi_{w,d^*} = \text{Encode}(\text{pk}_{w,d^*}, s),$$

where d^* is the value carried by the wire w and ψ_{w,d^*} is computed as in Dec. Formally, we proceed via induction on w to show that

$$\psi_{w,d^*} \in \Psi_{\text{pk}_{w,d^*}, s, j}.$$

where j is the depth of wire w . The base case $w = 1, \dots, \ell$ follows immediately from correctness of Encode. For the inductive step, consider a wire w at depth j for some gate $g = (u, v, w)$ where $u, v < w$. By the induction hypothesis,

$$\psi_{u,b^*} \in \Psi_{\text{pk}_{u,b^*}, s, j_0}, \quad \psi_{v,c^*} \in \Psi_{\text{pk}_{v,c^*}, s, j_1}$$

where $j_0, j_1 < j$ denote the depths of wires u and v respectively. It follows immediately from the correctness of Recode that

$$\psi_{w,d^*} \in \Psi_{\text{pk}_{w,d^*}, s, \max(j_0, j_1)+1} \subseteq \Psi_{\text{pk}_{w,d^*}, s, j}$$

which completes the inductive proof. Since $C(\text{ind}) = 1$ and $\text{pk}_{|C|,1} = \text{pk}_{\text{out}}$, we have $\psi_{|C|,1} \in \Psi_{\text{pk}_{\text{out}}, s, d_{\max}}$. Finally, by the correctness of (E, D) , $D(\psi_{|C|,1}, \tau) = m$. \square

6.3 Security

Lemma 6.4 (selective security). *For any adversary \mathcal{A} against selective security of the attribute-based encryption scheme, there exist an adversary \mathcal{B} against correlated pseudorandomness of TOR whose running time is essentially the same as that of \mathcal{A} , such that*

$$\text{Adv}_{\mathcal{A}}^{\text{PE}}(\lambda) \leq \text{Adv}_{\mathcal{B}}^{\text{CP}}(\lambda) + \text{negl}(\lambda)$$

where $\text{negl}(\lambda)$ captures the statistical security terms in TOR.

We begin by describing alternative algorithms, which would be useful later for constructing the adversary \mathcal{B} for the correlated pseudorandomness security game.

Alternative algorithms. Fix the selective challenge ind . We get from the “outside” the challenge $\text{pp}, (\text{pk}_i, \psi_i)_{i \in [\ell+1]}$ for correlated pseudorandomness. The main challenge is to design an alternative algorithm KeyGen^* for answering secret key queries without knowing $\text{sk}_{1,\text{ind}_1}, \dots, \text{sk}_{\ell,\text{ind}_\ell}$ or sk_{out} . The algorithm KeyGen^* will maintain the following invariant: on input C with $C(\text{ind}) = 0$,

- for every non-output wire $w = 1, \dots, |C| - 1$ carrying the value b^* , we will know $\text{sk}_{w,1-b^*}$ but not sk_{w,b^*} .

Moreover, we do not know $\text{sk}_{|C|,0}$ or $\text{sk}_{|C|,1} = \text{sk}_{\text{out}}$.

$\text{Setup}^*(\text{ind}, 1^\lambda, 1^\ell, d_{\max})$: Let

$$\begin{aligned} (\text{pk}_{i,1-\text{ind}_i}, \text{sk}_{i,1-\text{ind}_i}) &\leftarrow \text{Keygen}(\text{pp}) \text{ for } i \in [\ell] \\ \text{pk}_{\text{out}} &:= \text{pk}_{\ell+1} \\ \text{pk}_{i,\text{ind}_i} &:= \text{pk}_i \text{ for } i \in [\ell] \end{aligned}$$

$$\text{Output mpk} = \begin{pmatrix} \text{pk}_{1,0} & \text{pk}_{2,0} & \dots & \text{pk}_{\ell,0} & \\ \text{pk}_{1,1} & \text{pk}_{2,1} & \dots & \text{pk}_{\ell,1} & \text{pk}_{\text{out}} \end{pmatrix}$$

$\text{Enc}^*(\text{mpk}, \text{ind}, m)$: Set $\tau \leftarrow \text{E}(\psi_{\ell+1}, m)$ and output the ciphertext

$$\text{ct}_{\text{ind}} = (\psi_1, \psi_2, \dots, \psi_\ell, \tau)$$

where $\psi_1, \dots, \psi_{\ell+1}$ are provided in the challenge.

$\text{KeyGen}^*(\text{ind}, \text{msk}, C)$: where $C(\text{ind}) = 0$,

1. For each internal wire $w \in [\ell + 1, |C| - 1]$ of the circuit C carrying the value b^* for input ind , generate public/secret key pairs:

$$(\text{pk}_{w,1-b^*}, \text{sk}_{w,1-b^*}) \leftarrow \text{Keygen}(\text{pp})$$

We will generate pk_{w,b^*} using SimReKeyGen as described next.

2. For $w = \ell + 1, \dots, |C|$, let $g = (u, v, w)$ denote the gate for which w is the outgoing wire. Suppose wires u and v carry the values b^* and c^* , so that wire w carries the value $d^* := g_w(b^*, c^*)$. By the invariant above, we know $\text{sk}_{u,1-b^*}$ and $\text{sk}_{v,1-c^*}$ but not sk_{u,b^*} and sk_{v,c^*} . We start by generating

$$(\text{pk}_{w,d^*}, \text{rk}_{b^*,c^*}^w) \leftarrow \text{SimReKeyGen}(\text{pk}_{u,b^*}, \text{pk}_{v,c^*})$$

We generate the other three recoding keys using ReKeyGen as follows:

$$\begin{aligned} \text{rk}_{1-b^*,c^*}^w &\leftarrow \text{ReKeyGen}(\text{pk}_{u,1-b^*}, \text{pk}_{v,c^*}, \text{sk}_{u,1-b^*}, \text{pk}_{w,g_w(1-b^*,c^*)}) \\ \text{rk}_{b^*,1-c^*}^w &\leftarrow \text{ReKeyGen}(\text{pk}_{v,1-c^*}, \text{pk}_{u,b^*}, \text{sk}_{v,1-c^*}, \text{pk}_{w,g_w(b^*,1-c^*)}) \\ \text{rk}_{1-b^*,1-c^*}^w &\leftarrow \text{ReKeyGen}(\text{pk}_{u,1-b^*}, \text{pk}_{v,1-c^*}, \text{sk}_{u,1-b^*}, \text{pk}_{w,g_w(1-b^*,1-c^*)}) \end{aligned}$$

Note that $\text{rk}_{1-b^*,c^*}^w, \text{rk}_{1-b^*,1-c^*}^w$ are generated the same way in both KeyGen and KeyGen^* using $\text{sk}_{u,1-b^*}$.

Output the secret key

$$\text{sk}_C := \left(\text{rk}_{b,c}^w : w \in [\ell + 1, |C|], b, c \in \{0, 1\} \right)$$

Informally, the recoding key $\text{rk}_{b^*,1-c^*}^w$ looks the same as in KeyGen because of key indistinguishability, and rk_{b^*,c^*}^w (together with the simulated pk_{w,d^*}) looks the same as in KeyGen because of the recoding simulation property.

Game sequence. Next, consider the following sequence of games. We use $\text{Adv}_0, \text{Adv}_1, \dots$ to denote the advantage of the adversary \mathcal{A} in Games 0, 1, etc. Game 0 is the real experiment.

Game i for $i = 1, 2, \dots, q$. As in Game 0, except the challenger answers the first $i - 1$ key queries using KeyGen^* and the remaining $q - i$ key queries using KeyGen . For the i 'th key query C_i , we consider sub-Games $i.w$ as follows:

Game $i.w$, for $w = \ell + 1, \dots, |C_i|$. The challenger switches $(\text{rk}_{b,c}^w : b, c \in \{0, 1\})$ from KeyGen to KeyGen^* . More precisely:

- First, we switch $(\text{pk}_{w,d^*}, \text{rk}_{b^*,c^*}^w)$ from KeyGen to KeyGen^* . This relies on recoding simulation.
- Next, we switch $\text{rk}_{b^*,1-c^*}^w$ from KeyGen to KeyGen^* . This relies on key indistinguishability, w.r.t. sk_{b^*} and sk_{1-c^*} .
- The other two keys $\text{rk}_{1-b^*,c^*}^w, \text{rk}_{1-b^*,1-c^*}^w$ are generated the same way in both KeyGen and KeyGen^* .

By key indistinguishability and recoding simulation, we have

$$|\text{Adv}_{i,w} - \text{Adv}_{i,w+1}| \leq \text{negl}(\lambda) \text{ for all } i, w$$

Note that in Game q , the challenger runs Setup^* and answers all key queries using KeyGen^* with the selective challenge ind and generates the challenge ciphertext using Enc .

Game $q + 1$. Same as Game q , except the challenger generates the challenge ciphertext using Enc^* with $\psi_{\ell+1} = \text{Encode}(\text{pk}_{\ell+1}, s)$. Clearly,

$$\text{Adv}_{q+1} = \text{Adv}_q$$

Game $q + 2$. Same as Game $q + 1$, except $\psi_{\ell+1} \xleftarrow{\$} \mathcal{K}$. It is straight-forward to construct an adversary \mathcal{B} such that

$$|\text{Adv}_{q+1} - \text{Adv}_{q+2}| \leq \text{Adv}_{\mathcal{B}}^{\text{CP}}(\lambda)$$

Finally, $\text{Adv}_{q+2} \leq \text{negl}(\lambda)$ by the one-time semantic security of (E, D) . The lemma then follows readily.

7 Attribute-Based Encryption for Branching Programs

In this section, we present weak TOR and attribute-based encryption for branching programs, which capture the complexity class log-space. As noted in Section 2.2, we exploit the fact that in branching programs, the transition function depends on an input variable and the current state; this means that one of the two input encodings during recoding is always a “depth 0” encoding.

Branching programs. Recall that a branching program Γ is a directed acyclic graph in which every nonterminal node has exactly two outgoing edges labeled $(i, 0)$ and $(i, 1)$ for some $i \in [\ell]$. Moreover, there is a distinguished terminal accept node. Every input $x \in \{0, 1\}^\ell$ naturally induces a subgraph Γ_x containing exactly those edges labeled (i, x_i) . We say that Γ accepts x iff there is a path from the start node to the accept node in Γ_x . At the cost of possibly doubling the number of edges and vertices, we may assume that there is at most one edge connecting any two nodes in Γ .

7.1 Weak TOR

A weak “two-to-one” encoding (wTOR) scheme consists of the same algorithms as TOR, except that $\text{Keygen}(\text{pp}, j)$ takes an additional input $j \in \{0, 1\}$. That is, Keygen may produce different distribution of public/secret key pairs depending on j . Moreover, in ReKeyGen , the first public key is always generated using $\text{Keygen}(\text{pp}, 0)$ and the second using $\text{Keygen}(\text{pp}, 1)$; similarly, in Recode , the first encoding is always generated with respect to a public key from $\text{Keygen}(\text{pp}, 0)$ and the second from $\text{Keygen}(\text{pp}, 1)$. Similarly, the correctness and statistical security properties are relaxed.

Correctness. First, for every pk and $s \in \mathcal{S}$, there exists a family of sets $\Psi_{\text{pk}, s, j}, j = 0, 1, \dots, d_{\max}$:

- $\Psi_{\text{pk}, s, 1} \subseteq \dots \subseteq \Psi_{\text{pk}, s, d_{\max}}$.
- for all $\psi, \psi' \in \Psi_{\text{pk}, s, d_{\max}}$ and all $m \in \mathcal{M}$,

$$\text{D}(\psi', \text{E}(\psi, m)) = m$$

Secondly, the correctness of recoding requires that for any triple of key pairs $(\text{pk}_0, \text{sk}_0), (\text{pk}_1, \text{sk}_1), (\text{pk}_{\text{tgt}}, \text{sk}_{\text{tgt}})$ respectively in the support of $\text{Keygen}(\text{pp}, 0), \text{Keygen}(\text{pp}, 1), \text{Keygen}(\text{pp}, 1)$ and any encodings $\psi_0 \in \text{Encode}(\text{pk}_0, s)$ and $\psi_1 \in \Psi_{\text{pk}_1, s, j_1}$ where $0 < j_1$,

$$\text{Recode}(\text{rk}, \psi_0, \psi_1) \in \Psi_{\text{pk}_{\text{tgt}}, s, j_1+1}$$

Statistical Security Properties. We require recoding simulation as before, but not key indistinguishability. However, we require the following additional property:

(Back-tracking) : For all $(pk_0, sk_0) \leftarrow \text{Keygen}(pp, 0)$ and all $(pk_1, sk_1), (pk_{\text{tgt}}, sk_{\text{tgt}}) \leftarrow \text{Keygen}(pp, 1)$, the following distributions are identical:

$$\text{ReKeyGen}(pk_0, pk_1, sk_0, pk_{\text{tgt}}) \equiv -\text{ReKeyGen}(pk_0, pk_{\text{tgt}}, sk_0, pk_1)$$

Informally, this says that switching the order of pk_1 and pk_{tgt} as inputs to ReKeyGen is the same as switching the “sign” of the output. In our instantiations, the output of ReKeyGen lies in a group, so negating the output simply refers to applying the group inverse operation.

Remark 7.1. *Due to the additional back-tracking property, it is not the case that a TOR implies a weak TOR. However, we are able to instantiate weak TOR under weaker and larger classes of assumptions than TOR.*

Computational Security Property. We define the advantage function $\text{Adv}_{\mathcal{A}}^{\text{CP}}(\lambda)$ (modified to account for the additional input to Keygen) to be the absolute value of:

$$\Pr \left[\begin{array}{l} pp \leftarrow \text{Setup}(1^\lambda); s \leftarrow \mathcal{S}; \\ (pk_i, sk_i) \leftarrow \text{Keygen}(pp, 0), \\ \psi_i \leftarrow \text{Encode}(pk_i, s), i = 1, \dots, \ell; \\ (pk_{\ell+1}, sk_{\ell+1}) \leftarrow \text{Keygen}(pp, 1); \\ \psi'_0 \leftarrow \text{Encode}(pk_{\ell+1}, s); \\ b \xleftarrow{\$} \{0, 1\}; \psi'_1 \xleftarrow{\$} \mathcal{K} \\ b' \leftarrow \mathcal{A}(pk_1, \dots, pk_{\ell+1}, \psi_1, \dots, \psi_\ell, \psi'_b) \end{array} \right] - \frac{1}{2}$$

and we require that for all PPT \mathcal{A} , the advantage function $\text{Adv}_{\mathcal{A}}^{\text{CP}}(\lambda)$ is a negligible function in λ .

7.2 Weak TOR from LWE

We provide an instantiation of weak TOR from LWE. The main advantage over our construction of TOR in Section 5 is that the dependency of q on d_{\max} is linear in d_{\max} instead of exponential. Therefore, if q is quasi-polynomial, we can handle any polynomial d_{\max} , as opposed to an a-prior bounded d_{\max} .

Lemma 7.1. *Assuming $\text{dLWE}_{n,(\ell+2)m,q,\chi}$ where $q = O(d_{\max} n^3 \log n)$, there is a weak TOR scheme that is correct up to d_{\max} levels.*

Note that the parameters here are better than in Lemma 5.1. The construction of weak TOR from learning with errors follows:

- **Params** $(1^\lambda, d_{\max})$: First choose the LWE dimension $n = n(\lambda)$. Let the error distribution $\chi = \chi(n) = D_{\mathbb{Z}, \sqrt{n}}$, the error bound $B = B(n) = O(n)$, the modulus $q = q(n) = d_{\max} \cdot O(n^3 \log n)$, the number of samples $m = m(n) = O(n \log q)$ and the Gaussian parameter $s = s(n) = O(\sqrt{n \log q})$. Output the global public parameters $pp = (n, \chi, B, q, m, s)$. Define the domain \mathcal{S} of the encoding scheme to be \mathbb{Z}_q^n .

- **Keygen**(pp, j): Run the trapdoor generation algorithm $\text{TrapGen}(1^n, 1^m, q)$ to obtain a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ together with the trapdoor \mathbf{T} . Output

$$\text{pk} = \mathbf{A}; \quad \text{sk} = \mathbf{T}.$$

- **Encode**(\mathbf{A}, \mathbf{s}): Sample an error vector $\mathbf{e} \leftarrow \chi^m$ and output the encoding $\psi := \mathbf{A}^T \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^n$.
- **ReKeyGen**($\mathbf{A}_0, \mathbf{A}_1, \mathbf{A}_{\text{tgt}}, \mathbf{T}$): Outputs a low-norm matrix \mathbf{R} such that $\mathbf{A}_0 \mathbf{R} = \mathbf{A}_{\text{tgt}} - \mathbf{A}_1$. In particular,

$$\mathbf{R} \leftarrow \text{SampleD}(\mathbf{A}_0, \mathbf{T}_0, \mathbf{A}_{\text{tgt}} - \mathbf{A}_1, s)$$

- **SimReKeyGen**($\mathbf{A}_0, \mathbf{A}_1$): Sample a matrix $\mathbf{R} \leftarrow (D_{\mathbb{Z}, s})^{m \times m}$ by sampling each entry from the discrete Gaussian distribution $D_{\mathbb{Z}, s}$. Output

$$\text{rk} := \mathbf{R}; \quad \mathbf{A}_{\text{tgt}} := \mathbf{A}_0 \mathbf{R} + \mathbf{A}_1$$

- **Recode**($\text{rk}, \psi_0, \psi_1$): Outputs $\text{rk}^T \psi_0 + \psi_1$.

Correctness. We define the sets $\Psi_{\mathbf{A}, \mathbf{s}, j}$ for $\text{pk} := \mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \in \mathbb{Z}_q^n$ and $j \in [1 \dots d_{\max}]$ as follows:

$$\Psi_{\mathbf{A}, \mathbf{s}, j} = \{\mathbf{A}^T \mathbf{s} + \mathbf{e} : \|\mathbf{e}\|_\infty \leq B \cdot j \cdot (sm\sqrt{m})\}$$

The analysis is similar to that in the previous section. In particular, we observe right away that

- $\Psi_{\mathbf{A}, \mathbf{s}, 1} \subseteq \Psi_{\mathbf{A}, \mathbf{s}, 1} \subseteq \dots \subseteq \Psi_{\mathbf{A}, \mathbf{s}, d_{\max}}$.
- For any two encodings $\psi, \psi' \in \Psi_{\mathbf{A}, \mathbf{s}, d_{\max}}$ and $\mu \in \{0, 1\}^n$, $D(\psi', E(\psi, \mu)) = \mu$, as long as

$$B \cdot d_{\max} \cdot (sm\sqrt{m}) \leq q/4.$$

- Consider two encodings $\mathbf{A}^T \mathbf{s} + \mathbf{e} \in \text{Encode}(\mathbf{A}, \mathbf{s})$ and $\psi_1 \in \Psi_{\mathbf{A}_1, \mathbf{s}, j_1}$ for any $j_1 \in \mathbb{N}$. Then, $\psi_0 = \mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0$ and $\psi_1 := \mathbf{A}_1^T \mathbf{s} + \mathbf{e}_1$ where $\|\mathbf{e}_0\|_\infty \leq B$ and $\|\mathbf{e}_1\|_\infty \leq j_1 \cdot B \cdot (sm\sqrt{m})$.

Then, the recoded ciphertext ψ_{tgt} is computed as follows:

$$\begin{aligned} \psi_{\text{tgt}} &:= \mathbf{R}^T \psi_0 + \psi_1 \\ &= \mathbf{R}^T (\mathbf{A}_0^T \mathbf{s} + \mathbf{e}_0) + (\mathbf{A}_1^T \mathbf{s} + \mathbf{e}_1) \\ &= \mathbf{A}_{\text{tgt}}^T \mathbf{s} + \mathbf{e}_{\text{tgt}} \end{aligned}$$

where the last equation is because $\mathbf{A}_{\text{tgt}} = \mathbf{A}_0 \mathbf{R} + \mathbf{A}_1$ and we define $\mathbf{e}_{\text{tgt}} := \mathbf{R}^T \mathbf{e}_0 + \mathbf{e}_1$. Thus,

$$\begin{aligned} \|\mathbf{e}_{\text{tgt}}\|_\infty &\leq m \cdot \|\mathbf{R}\|_\infty \|\mathbf{e}_0\|_\infty + \|\mathbf{e}_1\|_\infty \\ &\leq m \cdot s\sqrt{m} \cdot B + B \cdot j_1 \cdot (sm\sqrt{m}) \\ &= (j_1 + 1) \cdot B \cdot (sm\sqrt{m}) \end{aligned}$$

exactly as required. Here, the second inequality is because $\|\mathbf{R}\|_\infty \leq s\sqrt{m}$ by Lemma 3.1. This finishes our proof of correctness.

Security. Correlated pseudorandomness follows from $\text{dLWE}_{n,(\ell+2)m,q,\chi}$ where $q = n \cdot d_{\max}$. Recoding simulation follows from Lemma 3.1 by an argument identical to the one for the construction of TOR in Section 5. For back-tracking, negation is simply the additive inverse over \mathbb{Z}_q^m .

7.3 Weak TOR from Bilinear Maps

We use asymmetric groups for maximal generality and for conceptual clarity. We consider cyclic groups G_1, G_2, G_T of prime order q and $e : G_1 \times G_2 \rightarrow G_T$ is a non-degenerate bilinear map. We require that the group operations in G and G_T as well the bilinear map e are computable in deterministic polynomial time with respect to λ . Let g_1, g_2 denote random generators of G_1, G_2 respectively. The DBDH Assumption says that, given $g_1, g_2, g_1^a, g_2^a, g_2^b$ and g_1^s , $e(g_1, g_2)^{abs}$ is pseudorandom.

- $\text{Params}(1^\lambda, d_{\max})$: Outputs $\text{pp} := (g_1, g_2, g_1^a, g_2^a)$.

- $\text{Keygen}(\text{pp}, j)$:

– If $j = 0$, then samples $t \xleftarrow{\$} \mathbb{Z}_q$ and outputs

$$(\text{pk}, \text{sk}) := ((g_1^{a/t}, g_2^{a/t}), t)$$

– If $j \geq 1$, output $\text{pk} \xleftarrow{\$} G_2$.

- $\text{Encode}(\text{pk}, s)$:

– If $\text{pk} = (g_1^{a/t}, g_2^{a/t}) \in G_1 \times G_2$, output $(g_1^{a/t})^s$

– If $\text{pk} \in G_2$, output $e(g_1^a, \text{pk})^s$

- $\text{Recode}(\text{rk}, c_0, c_1)$: Outputs $e(c_0, \text{rk}) \cdot c_1$.

- $\text{ReKeyGen}((g_1^{a/t}, g_2^{a/t}), \text{pk}_1, \text{pk}_{\text{tgt}}, t)$: Outputs $\text{rk} := (\text{pk}_{\text{tgt}} \cdot \text{pk}_1^{-1})^t \in G_2$.

- $\text{SimReKeyGen}((g_1^{a/t}, g_2^{a/t}), \text{pk}_1)$: Picks $z \xleftarrow{\$} \mathbb{Z}_q$ and outputs

$$\text{rk} := (g_2^{a/t})^z, \quad \text{pk}_{\text{tgt}} := \text{pk}_1 \cdot (g_2^a)^z$$

Correctness. Define $\Psi_{\text{pk},s,j} := \{\text{Encode}(\text{pk}, s)\}$. For recoding, observe that:

$$\begin{aligned} & \text{Recode}((\text{pk}_{\text{tgt}} \cdot \text{pk}_1^{-1})^t, g_1^{as/t}, e(g_1^a, \text{pk}_1)^s) \\ &= e(g_1^{as/t}, (\text{pk}_{\text{tgt}} \cdot \text{pk}_1^{-1})^t) \cdot e(g_1^a, \text{pk}_1)^s \\ &= e(g_1^a, (\text{pk}_{\text{tgt}} \cdot \text{pk}_1^{-1})^s) \cdot e(g_1^a, \text{pk}_1)^s \\ &= e(g_1^a, \text{pk}_{\text{tgt}})^s = \text{Encode}(\text{pk}_{\text{tgt}}, s) \end{aligned}$$

For back-tracking, negation is simply the multiplicative inverse over G_q .

Security. Correlation pseudorandomness follows readily from the DBDH assumption and its random self-reducibility.

7.4 Attribute-Based Encryption from weak TOR

$\text{Setup}(1^\lambda, 1^\ell, d_{\max})$: For each one of ℓ input bits, generate two public/secret key pairs. Also, generate a public/secret key pair for the start and accept states:

$$\begin{aligned} (\text{pk}_{i,b}, \text{sk}_{i,b}) &\leftarrow \text{Keygen}(\text{pp}, 0) \quad \text{for } i \in [\ell], b \in \{0, 1\} \\ (\text{pk}_{\text{start}}, \text{sk}_{\text{start}}) &\leftarrow \text{Keygen}(\text{pp}, 1) \\ (\text{pk}_{\text{accept}}, \text{sk}_{\text{accept}}) &\leftarrow \text{Keygen}(\text{pp}, 1) \end{aligned}$$

Output

$$\begin{aligned} \text{mpk} &:= \begin{pmatrix} \text{pk}_{1,0} & \text{pk}_{2,0} & \dots & \text{pk}_{\ell,0} & \text{pk}_{\text{start}} \\ \text{pk}_{1,1} & \text{pk}_{2,1} & \dots & \text{pk}_{\ell,1} & \text{pk}_{\text{accept}} \end{pmatrix} \\ \text{msk} &:= \begin{pmatrix} \text{sk}_{1,0} & \text{sk}_{2,0} & \dots & \text{sk}_{\ell,0} & \text{sk}_{\text{start}} \\ \text{sk}_{1,1} & \text{sk}_{2,1} & \dots & \text{sk}_{\ell,1} & \text{sk}_{\text{accept}} \end{pmatrix} \end{aligned}$$

$\text{Enc}(\text{mpk}, \text{ind}, m)$: For $\text{ind} \in \{0, 1\}^\ell$, choose a uniformly random $s \xleftarrow{\$} \mathcal{S}$ and encode it under the public keys specified by the index bits and the start state:

$$\begin{aligned} \psi_i &\leftarrow \text{Encode}(\text{pk}_{i,\text{ind}_i}, s) \quad \text{for all } i \in [\ell] \\ \psi_{\text{start}} &\leftarrow \text{Encode}(\text{pk}_{\text{start}}, s) \end{aligned}$$

Encrypt the message:

$$\tau \leftarrow \text{E}(\text{Encode}(\text{pk}_{\text{accept}}, s), m)$$

Output the ciphertext:

$$\text{ct}_{\text{ind}} = (\psi_1, \psi_2, \dots, \psi_\ell, \psi_{\text{start}}, \tau)$$

$\text{KeyGen}(\text{msk}, \Gamma)$: $\Gamma : \{0, 1\}^\ell \rightarrow \{0, 1\}$ is a branching program that takes a ℓ -bit input and outputs a single bit.

- For every node u , except the start and accept nodes, sample public/secret key pair:

$$(\text{pk}_u, \text{sk}_u) \leftarrow \text{Keygen}(\text{pp}, 1)$$

- For every edge (u, v) labeled (i, b) in Γ , sample a recoding key $\text{rk}_{u,v}$ as follows:

$$\text{rk}_{u,v} \leftarrow \text{ReKeyGen}(\text{pk}_{i,b}, \text{pk}_u, \text{sk}_{i,b}, \text{pk}_v)$$

The secret key sk_Γ is the collection of all the recoding keys $\text{rk}_{u,v}$ for every edge (u, v) in Γ .

$\text{Dec}(\text{sk}_\Gamma, \text{ct}_{\text{ind}})$: Suppose $\Gamma(\text{ind}) = 1$; output \perp otherwise. Let Π denote the (directed) path from the start node to the accept node in Γ_{ind} . For every edge (u, v) labeled (i, ind_i) in Π , apply the recoding algorithm on the two encodings ψ_i, ψ_u and the recoding key $\text{rk}_{u,v}$:

$$\psi_v \leftarrow \text{Recode}(\text{rk}_{u,v}, \psi_i, \psi_u)$$

If $\Gamma(\text{ind}) = 1$, we obtain ψ_{accept} . Decrypt and output the message:

$$m \leftarrow \text{D}(\psi_{\text{accept}}, \tau)$$

7.4.1 Correctness

Lemma 7.2 (correctness). *Let $\mathcal{G} = \{\Gamma\}_\lambda$ be a collection of polynomial-size branching programs of depth at most d_{\max} and let wTOR be a weak “two-to-one” recoding scheme for d_{\max} levels. Then, the construction presented above is a correct attribute-based encryption scheme for \mathcal{G} .*

Proof. Let Π denote the directed path from the start to the accept nodes in Γ_{ind} . We show via induction on nodes v along the path Π that

$$\psi_v \in \Psi_{\text{pk}_v, s, j}$$

where j is the depth of node v along the path. The base case for $v := \text{start node}$ follows immediately from correctness of **Encode**. For the inductive step, consider a node v along the path Π at depth j for some edge (u, v) labeled (i, ind_i) . By the induction hypothesis,

$$\psi_u \in \Psi_{\text{pk}_u, s, j_0}$$

where $j_0 < j$ denote the depths of node u . Also by the correctness of the **Encode** algorithm, for all $i \in [\ell]$

$$\psi_i \in \Psi_{\text{pk}_{i, \text{ind}_i}, s, 0}$$

It follows immediately from the correctness of **Recode** that

$$\psi_v \in \Psi_{\text{pk}_v, s, j_0+1} \subseteq \Psi_{\text{pk}_v, s, j}$$

which completes the inductive proof. Since $C(\text{ind}) = 1$, we have

$$\psi_{\text{accept}} \in \Psi_{\text{pk}_{\text{accept}}, s, d_{\max}}$$

Recall that $\tau \leftarrow \text{E}(\text{Encode}(\text{pk}_{\text{accept}}, s), m)$. Finally, by the correctness of (E, D) ,

$$\text{D}(\psi_{\text{accept}}, \tau) = m \quad \square$$

7.4.2 Selective Security

Lemma 7.3 (selective security). *For any adversary \mathcal{A} against selective security of the attribute-based encryption scheme for branching programs, there exist an adversary \mathcal{B} against correlated pseudorandomness of wTOR whose running time is essentially the same as that of \mathcal{A} , such that*

$$\text{Adv}_{\mathcal{A}}^{\text{PE}}(\lambda) \leq \text{Adv}_{\mathcal{B}}^{\text{CP}}(\lambda) + \text{negl}(\lambda)$$

where $\text{negl}(\lambda)$ captures the statistical security terms in TOR.

In the proof of security, we will rely crucially on the following combinatorial property of branching programs: for any input x , the graph Γ_x does not contain any cycles as an *undirected* graph.

Alternative algorithms. Fix the selective challenge ind . We also get a collection of public keys, corresponding encodings from the “outside”: $(\text{pk}_i, \psi_i)_{i \in [\ell+2]}$, where the challenge is to decide whether $\psi_{\ell+1}$ is $\text{Encode}(\text{pk}_{\ell+2}, s)$ or random. The main challenge is design an alternative algorithm KeyGen^* for answering secret key queries without knowing $\text{sk}_{1,\text{ind}_1}, \dots, \text{sk}_{\ell,\text{ind}_\ell}$ or $\text{sk}_{\text{start}}, \text{sk}_{\text{accept}}$. We consider the following “alternative” algorithms.

$\text{Setup}^*(1^\lambda, 1^\ell, d_{\max})$: Let

$$\begin{aligned} (\text{pk}_{i,1-\text{ind}_i}, \text{sk}_{i,1-\text{ind}_i}) &\leftarrow \text{Keygen}(\text{pp}, 0) \text{ for } i \in [\ell] \\ \text{pk}_{i,\text{ind}_i} &:= \text{pk}_i \text{ for } i \in [\ell] \\ \text{pk}_{\text{start}} &:= \text{pk}_{\ell+1} \\ \text{pk}_{\text{accept}} &:= \text{pk}_{\ell+2} \end{aligned}$$

Define and output the master public key as follows:

$$\text{mpk} = \begin{pmatrix} \text{pk}_{1,0} & \text{pk}_{2,0} & \dots & \text{pk}_{\ell,0} & \text{pk}_{\text{start}} \\ \text{pk}_{1,1} & \text{pk}_{2,1} & \dots & \text{pk}_{\ell,1} & \text{pk}_{\text{accept}} \end{pmatrix}$$

$\text{Enc}^*(\text{mpk}, \text{ind}, m)$: Define

$$\begin{aligned} \psi_{i,\text{ind}_i} &:= \psi_i \quad \text{for all } i \in [\ell] \\ \psi_{\text{start}} &:= \psi_{\ell+1} \\ \psi_{\text{accept}} &:= \psi_{\ell+2} \end{aligned}$$

Encrypt the message m :

$$\tau \leftarrow \text{E}(\psi_{\text{accept}}, b)$$

Output the simulated ciphertext

$$\text{ct}_{\text{ind}} = (\psi_1, \psi_2, \dots, \psi_\ell, \psi_{\text{start}}, \tau)$$

$\text{KeyGen}^*(\text{msk}, \Gamma)$: Let Γ'_{ind} denote the *undirected* graph obtained from Γ_{ind} by treating every directed edge as an undirected edge (while keeping the edge label). Observe that Γ'_{ind} satisfies the following properties:

- Γ'_{ind} contains no cycles. This is because Γ_{ind} is acyclic and every nonterminal node contains exactly one outgoing edge.
- The start node and the accept node lie in different connected components in Γ'_{ind} , since $\Gamma(\text{ind}) = 0$.

Simulation invariant: for each “active” edge labeled (i, ind_i) from node u to node v , simulate the recoding key. Choose our own public/secret key pair for each “inactive” edges $(i, 1 - \text{ind}_i)$ and generate the recoding key honestly.

- Run a DFS in Γ'_{ind} starting from the start node. Whenever we visit a new node v from a node u along an edge labeled (i, ind_i) , we set:

$$\begin{aligned} (\text{pk}_v, \text{rk}_{u,v}) &\leftarrow \text{SimReKeyGen}(\text{pk}_{i,\text{ind}}, \text{pk}_u) & \text{if } (u, v) \text{ is a directed edge in } \Gamma \\ (\text{pk}_v, -\text{rk}_{v,u}) &\leftarrow \text{SimReKeyGen}(\text{pk}_{i,\text{ind}}, \text{pk}_u) & \text{if } (v, u) \text{ is a directed edge in } \Gamma \end{aligned}$$

Here, we exploit the back-tracking property in wTOR.

Note that since $\Gamma(\text{ind}) = 0$, then the accept node is not assigned a public key by this process.

- For all nodes u without an assignment, run $(\text{pk}_u, \text{sk}_u) \leftarrow \text{Keygen}(\text{pp}, 1)$.
- For every remaining edge (u, v) labeled $(i, 1 - \text{ind}_i)$ in Γ , sample a recoding key $\text{rk}_{u,v}$ as in KeyGen using $\text{sk}_{i,1-\text{ind}}$ as follows:

$$\text{rk}_{u,v} \leftarrow \text{ReKeyGen}(\text{pk}_{i,1-\text{ind}}, \text{pk}_u, \text{sk}_{i,1-\text{ind}}, \text{pk}_v)$$

The secret key sk_Γ is simply the collection of all the recoding keys $\text{rk}_{u,v}$ for every edge (u, v) in Γ .

Game sequence. Next, consider the following sequence of games. We use $\text{Adv}_0, \text{Adv}_1, \dots$ to denote the advantage of the adversary \mathcal{A} in Games 0, 1, etc. Let n denote the number of edges in a branching program Γ labeled (i, ind_i) for some i , and for all $j \in [n]$ let e_j denote the actual edge.

Game 0. Real experiment.

Game i for $i = 1, 2, \dots, q$. As in Game 0, except the challenger answers the first $i - 1$ key queries using KeyGen^* and the remaining $q - i$ key queries using KeyGen . For the i 'th key query Γ_i , we consider sub-Games $i.e$ as follows:

Game $i.j$, for $j = 1, \dots, n$. For edge $e_j = (u, v)$ labeled (i, ind_i) , the challenger switches the simulated recoding key $\text{rk}_{u,v}$ from KeyGen to KeyGen^* . We rely on recoding simulation and back-tracking properties simultaneously.

By recoding simulation and back-tracking, we have:

$$|\text{Adv}_{i,e} - \text{Adv}_{i,e+1}| \leq \text{negl}(\lambda) \text{ for all } i, e$$

Note that in Game q , the challenger runs Setup^* and answers all key queries using KeyGen^* with the selective challenge ind and generates the challenge ciphertext using Enc .

Game $q + 1$. Same as Game q , except the challenger generates the challenge ciphertext using Enc^* with $\psi_{\ell+2} \leftarrow \text{Encode}(\text{pk}_{\ell+2}, s)$.

$$\text{Adv}_{q+1} = \text{Adv}_q$$

Game $q + 2$. Same as Game $q + 1$, except $\psi_{\ell+2} \xleftarrow{\$} \mathcal{K}$. It is straight-forward to construct an adversary \mathcal{B} such that

$$|\text{Adv}_{q+1} - \text{Adv}_{q+2}| \leq \text{Adv}_{\mathcal{B}}^{\text{CP}}(\lambda)$$

Finally, $\text{Adv}_{q+2} \leq \text{negl}(\lambda)$ by the one-time semantic security of (E, D) . The lemma then follows readily.

Acknowledgments.

We thank Dan Boneh and Shafi Goldwasser for helpful comments and insightful conversations.

References

- [ABB10a] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, pages 553–572, 2010.
- [ABB10b] Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In *CRYPTO*, pages 98–115, 2010.
- [ABV⁺12] Shweta Agrawal, Xavier Boyen, Vinod Vaikuntanathan, Panagiotis Voulgaris, and Hoeteck Wee. Functional encryption for threshold functions (or, fuzzy IBE) from lattices. In *Public Key Cryptography*, pages 280–297, 2012.
- [AFGH06] Giuseppe Ateniese, Kevin Fu, Matthew Green, and Susan Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.
- [AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, pages 21–40, 2011.
- [AIK10] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. From secrecy to soundness: Efficient verification via secure computation. In *ICALP (1)*, pages 152–163, 2010.
- [Ajt99] Miklós Ajtai. Generating hard instances of the short basis problem. In *ICALP*, pages 1–9, 1999.
- [AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.
- [BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.
- [BBS98] Matt Blaze, Gerrit Bleumer, and Martin Strauss. Divertible protocols and atomic proxy cryptography. In *EUROCRYPT*, pages 127–144, 1998.
- [BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In *CRYPTO*, pages 213–229, 2001.
- [BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In *ACM CCS*, pages 784–796, 2012.
- [Boy13] Xavier Boyen. Attribute-based functional encryption on lattices. In *TCC*, pages 122–142, 2013.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.
- [CHK09] David Cash, Dennis Hofheinz, and Eike Kiltz. How to delegate a lattice basis. Cryptology ePrint Archive, Report 2009/351, 2009.
- [CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012.
- [CKV10] Kai-Min Chung, Yael Kalai, and Salil P. Vadhan. Improved delegation of computation using fully homomorphic encryption. In *CRYPTO*, pages 483–501, 2010.
- [Coc01] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [GGH12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610, 2012.
- [GGH⁺13] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2013/128, 2013.
- [GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.
- [GHW11] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of a ciphertext. In *Proceedings of the 20th USENIX conference on Security, SEC’11*, pages 34–34, Berkeley, CA, USA, 2011. USENIX Association.
- [GKP⁺13] Shafi Goldwasser, Yael Kalai, Raluca Ada Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Succinct functional encryption and its power: Reusable garbled circuits and beyond. In *STOC*, 2013. To appear.
- [GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, pages 113–122, 2008.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, pages 89–98, 2006.
- [GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.
- [GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.

- [HRSV11] Susan Hohenberger, Guy N. Rothblum, Abhi Shelat, and Vinod Vaikuntanathan. Securely obfuscating re-encryption. *J. Cryptology*, 24(4):694–719, 2011.
- [LOS⁺10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.
- [LW10] Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In *TCC*, pages 455–479, 2010.
- [LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, pages 180–198, 2012.
- [Mic00] Silvio Micali. Computationally sound proofs. *SIAM J. Comput.*, 30(4):1253–1298, 2000.
- [MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, pages 700–718, 2012.
- [MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on voronoi cell computations. In *STOC*, pages 351–358, 2010.
- [O’N10] Adam O’Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010.
- [OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.
- [PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC*, pages 422–439, 2012.
- [Reg09] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009.
- [RS10] Alon Rosen and Gil Segev. Chosen-ciphertext security via correlated products. *SIAM J. Comput.*, 39(7):3058–3088, 2010.
- [Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
- [SS10a] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM CCS*, pages 463–472, 2010.
- [SS10b] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *ASIACRYPT*, pages 377–394, 2010.

- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
- [Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, pages 619–636, 2009.
- [Wat12] Brent Waters. Functional encryption for regular languages. In *CRYPTO*, pages 218–235, 2012.

A Extensions

A.1 Outsourcing Decryption

In this section we show how to modify our main construction of attribute-based encryption to support outsourcing of decryption circuits, similar to [GHW11]. We require that the Keygen algorithm returns two keys:

- the evaluation key ek_C , that is given to a computationally powerful proxy,
- and a decryption key dk , given to the client.

Given a ciphertext ct_{ind} , the proxy must perform the “bulk” of the computation and return a new ciphertext ct'_{ind} that is forwarded to the client. Using the decryption key dk , the client can decrypt and learn the message m iff the predicate $C(\text{ind})$ is satisfied. We emphasize that that amount of computation the client needs to perform to decrypt the message must be independent on the circuit size. Intuitively, the security ensures that an adversary should learn nothing about the message, conditioned on that it queries for decryption keys dk 's for predicates that are not satisfied by the challenge index (note, the adversary *can* query for evaluation keys separately for predicates that are satisfied).

Intuitively, we modify the main construction as follows. As before, the key-generation algorithm assigns two keys for each circuit wire. The evaluation key consists of all the recoding keys for the circuit. In addition, the output wire has another key pk_{out} which now plays a special role. The recoding key from $\text{pk}_{|C|,1}$ to pk_{out} is only given to the client as the decryption key. If $C(\text{ind}) = 1$, the proxy computes an encoding under the $\text{pk}_{|C|,1}$ and forwards it to the client. The client applies the transformation, and decrypts the message. For technical reasons, since we are using “two-to-one” recoding mechanism, we need to introduce an auxiliary public key pk_{in} and a corresponding encoding.

Setup($1^\lambda, 1^\ell, d_{\text{max}}$) : For each of the ℓ input wires, generate two public/secret key pairs. Also, generate an additional public/secret key pair:

$$\begin{aligned} (\text{pk}_{i,b}, \text{sk}_{i,b}) &\leftarrow \text{Keygen}(\text{pp}) \quad \text{for } i \in [\ell], b \in \{0, 1\} \\ (\text{pk}_{\text{out}}, \text{sk}_{\text{out}}) &\leftarrow \text{Keygen}(\text{pp}) \\ (\text{pk}_{\text{in}}, \text{sk}_{\text{in}}) &\leftarrow \text{Keygen}(\text{pp}) \end{aligned}$$

Output

$$\text{mpk} := \begin{pmatrix} \text{pk}_{1,0} & \text{pk}_{2,0} & \dots & \text{pk}_{\ell,0} & \text{pk}_{\text{in}} \\ \text{pk}_{1,1} & \text{pk}_{2,1} & \dots & \text{pk}_{\ell,1} & \text{pk}_{\text{out}} \end{pmatrix} \quad \text{msk} := \begin{pmatrix} \text{sk}_{1,0} & \text{sk}_{2,0} & \dots & \text{sk}_{\ell,0} & \text{sk}_{\text{in}} \\ \text{sk}_{1,1} & \text{sk}_{2,1} & \dots & \text{sk}_{\ell,1} & \text{sk}_{\text{out}} \end{pmatrix}$$

Enc($\text{mpk}, \text{ind}, m$) : For $\text{ind} \in \{0, 1\}^\ell$, choose a uniformly random $s \xleftarrow{\$} \mathcal{S}$ and encode it under the public keys specified by the index bits:

$$\psi_i \leftarrow \text{Encode}(\text{pk}_{i,\text{ind}_i}, s) \text{ for all } i \in [\ell]$$

Encode s under the input public key:

$$\psi_{\text{in}} \leftarrow \text{Encode}(\text{pk}_{\text{in}}, s)$$

Encrypt the message m :

$$\tau \leftarrow E(\text{Encode}(\text{pk}_{\text{out}}, s), m)$$

Output the ciphertext

$$\text{ct}_{\text{ind}} := (\psi_1, \psi_2, \dots, \psi_\ell, \psi_{\text{in}}, \tau)$$

KeyGen(msk, C) :

1. For every non-input wire $w = \ell + 1, \dots, |C|$ of the circuit C , and every $b \in \{0, 1\}$, generate public/secret key pairs:

$$(\text{pk}_{w,b}, \text{sk}_{w,b}) \leftarrow \text{Keygen}(\text{pp})$$

2. For the gate $g = (u, v, w)$ with output wire w , compute the four recoding keys $\text{rk}_{b,c}^w$ (for $b, c \in \{0, 1\}$):

$$\text{rk}_{b,c}^w \leftarrow \text{ReKeyGen}(\text{pk}_{u,b}, \text{pk}_{v,c}, \text{sk}_{u,b}, \text{pk}_{w,g_w(b,c)})$$

3. Also, compute the recoding key

$$\text{rk}^{\text{out}} \leftarrow \text{ReKeyGen}(\text{pk}_{|C|,1}, \text{pk}_{\text{in}}, \text{sk}_{|C|,1}, \text{pk}_{\text{out}})$$

Output the evaluation key which is a collection of $4(|C| - \ell)$ recoding keys

$$\text{ek}_C := (\text{rk}_{b,c}^w : w \in [\ell + 1, |C|], b, c \in \{0, 1\})$$

and the decryption key $\text{dk} := \text{rk}^{\text{out}}$.

Eval($\text{ek}_C, \text{ct}_{\text{ind}}$) : We tacitly assume that ct_{ind} contains the index ind . For $w = \ell + 1, \dots, |C|$, let $g = (u, v, w)$ denote the gate with output wire w . Suppose wires u and v carry the values b^* and c^* , so that wire w carries the value $d^* := g_w(b^*, c^*)$. Compute

$$\psi_{w,d^*} \leftarrow \text{Recode}(\text{rk}_{b^*,c^*}^w, \psi_{u,b^*}, \psi_{v,c^*})$$

If $C(\text{ind}) = 1$, then we would have computed $\psi_{|C|,1}$. Output

$$\text{ct}'_{\text{ind}} := (\psi_{|C|,1}, \psi_{\text{in}}, \tau)$$

If $C(\text{ind}) = 0$, output \perp .

Dec($\text{dk}, \text{ct}'_{\text{ind}}$) : Apply the transformation

$$\psi_{\text{out}} \leftarrow \text{Recode}(\text{rk}^{\text{out}}, \psi_{\text{in}}, \psi_{|C|,1})$$

and output the message

$$m \leftarrow D(\psi_{\text{out}}, \tau)$$

Security. We informally state how to modify the simulator in the proof of security in Section-6.4. The simulator gets $\{\text{pk}_i, \psi_i\}_{i \in [\ell+2]}$ from the “outside”. It assigns $\text{pk}_1, \dots, \text{pk}_\ell$ as the public keys specified by the bits of ind and $\text{pk}_{\text{in}} := \text{pk}_{\ell+1}, \text{pk}_{\text{out}} := \text{pk}_{\ell+2}$. It is easy to see how to simulate the ciphertext: all the input encodings become a part of it, as well as an encryption of the message using $\psi_{\text{out}} := \psi_{\ell+2}$. Now, the evaluation key ek is simulated by applying the TOR simulator.

- For query C such that $C(\text{ind}) = 0$, the simulator can choose $(\text{pk}_{|C|,1}, \text{sk}_{|C|,1})$ by itself (the public key $\text{pk}_{|C|,0}$ is “fixed” by the TOR simulator). Hence, the decryption key dk can be computed using $\text{sk}_{|C|,1}$.
- On the other hand, for query C such that $C(\text{ind}) = 1$, the adversary is not allowed to obtain the decryption key dk , hence there is not need to simulate it.

A.2 Extending Secret Keys

Consider the following problem: a users holds two (or more) secret keys sk_{C_1} and sk_{C_2} . C_1 allows to decrypt all ciphertexts addressed to *human resources* department and C_2 allows to decrypt ciphertexts addressed to *share holders*. The user wishes to create (and delegate) another secret key sk_{C^*} that allows to decrypt ciphertexts addressed to *human resources* and *share holders*. The question that we study is whether it is possible to allow the user to compute sk_{C^*} without calling the authority holding the master secret key msk . More formally, given $\{\text{sk}_{C_i}\}_{i \in [q]}$ a users should be able to compute a secret key sk_{C^*} for any circuit C^* that is an black-box monotone composition of C_i ’s. Note that only monotone compositions are realizable, since otherwise a users holding a secret keys sk_{C_1} where $C_1(\text{ind}) = 0$ could come up with a secret key for $\overline{C_1}$ and hence break any notion of security.

To suppose monotone extensions, it is enough to show how to obtain (1) $\text{sk}_{C_1 \text{ AND } C_2}$ given $\text{sk}_{C_1}, \text{sk}_{C_2}$, and (2) $\text{sk}_{C_1 \text{ OR } C_2}$ given $\text{sk}_{C_1}, \text{sk}_{C_2}$. We start from the construction presented in Section-A.1. We note that the security of that construction does not break if we give the secret key associated with the output value 0 ($\text{sk}_{|C_i|,1}$) as a part of the secret key sk_{C_i} . This is because our simulation proceeds by sampling $(\text{pk}_{|C_i|,1}, \text{sk}_{|C_i|,1})$ honestly using **Keygen** algorithm and the fact the adversary is restricted to quires C_i such that $C_i(\text{ind}) = 0$. Hence, given $\text{sk}_{|C_1|,1}$ and $\text{sk}_{|C_2|,1}$, let $C^* = C_1 \text{ AND } C_2$. The user computes sk_{C^*} as $(\text{ek}_{C_1}, \text{ek}_{C_2})$ plus four recoding keys $\text{rk}_{b,c}^{C^*}$ (for $b, c \in \{0, 1\}$):

$$\begin{aligned} (\text{pk}_{|C^*|,0}, \text{rk}_{0,0}^{C^*}) &\leftarrow \text{SimReKeyGen}(\text{pk}_{|C_1|,0}, \text{pk}_{|C_2|,0}) \\ \text{rk}_{0,1}^{C^*} &\leftarrow \text{ReKeyGen}\left(\text{pk}_{|C_1|,0}, \text{pk}_{|C_2|,1}, \text{sk}_{|C_2|,1}, \text{pk}_{|C^*|,0}\right) \\ \text{rk}_{1,0}^{C^*} &\leftarrow \text{ReKeyGen}\left(\text{pk}_{|C_1|,1}, \text{pk}_{|C_2|,0}, \text{sk}_{|C_1|,1}, \text{pk}_{|C^*|,0}\right) \\ \text{rk}_{1,1}^{C^*} &\leftarrow \text{ReKeyGen}\left(\text{pk}_{|C_1|,1}, \text{pk}_{|C_2|,1}, \text{sk}_{|C_1|,1}, \text{pk}_{\text{out}}\right) \end{aligned}$$

As before, the message is encrypted under the encoding $\psi_{\text{out}} \leftarrow \text{Encode}(\text{pk}_{\text{out}}, s)$. The construction extends similarly to support **OR** compositions. Furthermore, arbitrary monotone structures can be realized by sampling keys associated with value 1 (pk_1, sk_1) honestly and computing the recoding keys as above, until the final wire is assigned to pk_{out} .

Fully Key-Homomorphic Encryption, Arithmetic Circuit ABE, and Compact Garbled Circuits*

Dan Boneh[†] Craig Gentry[‡] Sergey Gorbunov[§] Shai Halevi[¶]
Valeria Nikolaenko^{||} Gil Segev^{**} Vinod Vaikuntanathan^{††}
Dhinakaran Vinayagamurthy^{‡‡}

May 20, 2014

Abstract

We construct the first (key-policy) attribute-based encryption (ABE) system with short secret keys: the size of keys in our system depends only on the depth of the policy circuit, not its size. Our constructions extend naturally to arithmetic circuits with arbitrary fan-in gates thereby further reducing the circuit depth. Building on this ABE system we obtain the first reusable circuit garbling scheme that produces garbled circuits whose size is the same as the original circuit *plus* an additive $\text{poly}(\lambda, d)$ bits, where λ is the security parameter and d is the circuit depth. Save the additive $\text{poly}(\lambda, d)$ factor, this is the best one could hope for. All previous constructions incurred a *multiplicative* $\text{poly}(\lambda)$ blowup. As another application, we obtain (single key secure) functional encryption with short secret keys.

We construct our attribute-based system using a mechanism we call *fully key-homomorphic encryption* which is a public-key system that lets anyone translate a ciphertext encrypted under a public-key \mathbf{x} into a ciphertext encrypted under the public-key $(f(\mathbf{x}), f)$ of the same plaintext, for any efficiently computable f . We show that this mechanism gives an ABE with short keys. Security is based on the subexponential hardness of the learning with errors problem.

We also present a second (key-policy) ABE, using multilinear maps, with short ciphertexts: an encryption to an attribute vector \mathbf{x} is the size of \mathbf{x} plus $\text{poly}(\lambda, d)$ additional bits. This gives a reusable circuit garbling scheme where the size of the garbled input is short, namely the same as that of the original input, *plus* a $\text{poly}(\lambda, d)$ factor.

*This is the full version of a paper that appeared in Eurocrypt 2014 [BGG⁺14]. This work is a merge of two closely related papers [GGH⁺13d, BNS13].

[†]Stanford University, Stanford, CA, USA. Email: dabo@cs.stanford.edu.

[‡]IBM Research, Yorktown, NY, USA. Email: cbgentry@us.ibm.com.

[§]MIT, Cambridge, MA, USA. Email: sergeyg@mit.edu. This work was partially done while visiting IBM T. J. Watson Research Center.

[¶]IBM Research, Yorktown, NY, USA. Email: shaih@alum.mit.edu.

^{||}Stanford University, Stanford, CA, USA. Email: valerini@cs.stanford.edu.

^{**}Hebrew University, Jerusalem, Israel. Email: segev@cs.huji.ac.il. This work was partially done while the author was visiting Stanford University.

^{††}MIT, Cambridge, MA, USA. Email: vinodv@csail.mit.edu.

^{‡‡}University of Toronto, Toronto, Ontario, Canada. Email: dhinakaran5@cs.toronto.edu.

1 Introduction

(Key-policy) attribute-based encryption [SW05, GPSW06] is a public-key encryption mechanism where every secret key sk_f is associated with some function $f : \mathcal{X} \rightarrow \mathcal{Y}$ and an encryption of a message μ is labeled with a public attribute vector $\mathbf{x} \in \mathcal{X}$. The encryption of μ can be decrypted using sk_f only if $f(\mathbf{x}) = 0 \in \mathcal{Y}$. Intuitively, the security requirement is collusion resistance: a coalition of users learns nothing about the plaintext message μ if none of their individual keys are authorized to decrypt the ciphertext.

Attribute-based encryption (ABE) is a powerful generalization of identity-based encryption [Sha84, BF03, Coc01] and fuzzy IBE [SW05, ABV⁺12] and is a special case of functional encryption [BSW11]. It is used as a building-block in applications that demand complex access control to encrypted data [PTMW06], in designing protocols for verifiably outsourcing computations [PRV12], and for single-use functional encryption [GKP⁺13b]. Here we focus on key-policy ABE where the access policy is embedded in the secret key. The dual notion called ciphertext-policy ABE can be realized from this using universal circuits, as explained in [GPSW06, GGH⁺13c].

The past few years have seen much progress in constructing secure and efficient ABE schemes from different assumptions and for different settings. The first constructions [GPSW06, LOS⁺10, OT10, LW12, Wat12, Boy13, HW13] apply to predicates computable by Boolean formulas which are a subclass of log-space computations. More recently, important progress has been made on constructions for the set of all polynomial-size circuits: Gorbunov, Vaikuntanathan, and Wee [GVW13] gave a construction from the Learning With Errors (LWE) problem and Garg, Gentry, Halevi, Sahai, and Waters [GGH⁺13c] gave a construction using multilinear maps. In both constructions the policy functions are represented as Boolean circuits composed of fan-in 2 gates and the secret key size is proportional to the *size* of the circuit.

Our results. We present two new key-policy ABE systems. Our first system, which is the centerpiece of this paper, is an ABE based on the learning with errors problem [Reg05] that supports functions f represented as arithmetic circuits with large fan-in gates. It has secret keys whose size is proportional to *depth* of the circuit for f , not its size. Secret keys in previous ABE constructions contained an element (such as a matrix) for every gate or wire in the circuit. In our scheme the secret key is a single matrix corresponding only to the final output wire from the circuit. We prove *selective* security of the system and observe that by a standard complexity leveraging argument (as in [BB11]) the system can be made adaptively secure.

Theorem 1.1 (Informal). *Let λ be the security parameter. Assuming subexponential LWE, there is an ABE scheme for the class of functions with depth- d circuits where the size of the secret key for a circuit C is $\text{poly}(\lambda, d)$.*

Our second ABE system, based on multilinear maps ([BS02],[GGH13a]), optimizes the ciphertext size rather than the secret key size. The construction here relies on a generalization of broadcast encryption [FN93, BGW05, BW13] and the attribute-based encryption scheme of [GGH⁺13c]. Previously, ABE schemes with short ciphertexts were known only for the class of Boolean formulas [ALdP11].

Theorem 1.2 (Informal). *Let λ be the security parameter. Assuming that d -level multilinear maps exist, there is an ABE scheme for the class of functions with depth- d circuits where the size of the encryption of an attribute vector \mathbf{x} is $|\mathbf{x}| + \text{poly}(\lambda, d)$.*

Our ABE schemes result in a number of applications and have many desirable features, which we describe next.

Applications to reusable garbled circuits. Over the years, garbled circuits and variants have found many uses: in two party [Yao86] and multi-party secure protocols [GMW87, BMR90], one-time programs [GKR08], key-dependent message security [BH10], verifiable computation [GGP10], homomorphic computations [GHV10] and many others. Classical circuit garbling schemes produced single-use garbled circuits which could only be used in conjunction with one garbled input. Goldwasser et al. [GKP⁺13b] recently showed the first fully reusable circuit garbling schemes and used them to construct token-based program obfuscation schemes and k -time programs [GKP⁺13b].

Most known constructions of both single-use and reusable garbled circuits proceed by garbling each gate to produce a garbled truth table, resulting in a *multiplicative* size blowup of $\text{poly}(\lambda)$. A fundamental question regarding garbling schemes is: *How small can the garbled circuit be?*

There are three exceptions to the gate-by-gate garbling method that we are aware of. The first is the “free XOR” optimization for *single-use* garbling schemes introduced by Kolesnikov and Schneider [KS08] where one produces garbled tables only for the AND gates in the circuit C . This still results in a multiplicative $\text{poly}(\lambda)$ overhead but proportional to the number of AND gates (as opposed to the total number of gates). Secondly, Lu and Ostrovsky [LO13] recently showed a *single-use* garbling scheme for RAM programs, where the size of the garbled program grows as $\text{poly}(\lambda)$ times its running time. Finally, Goldwasser et al. [GKP⁺13a] show how to (reusably) garble non-uniform Turing machines under a non-standard and non-falsifiable assumption and incurring a multiplicative $\text{poly}(\lambda)$ overhead in the size of the non-uniformity of the machine. In short, all known garbling schemes (even in the single-use setting) suffer from a multiplicative overhead of $\text{poly}(\lambda)$ in the circuit size or the running time.

Using our first ABE scheme (based on LWE) in conjunction with the techniques of Goldwasser et al. [GKP⁺13b], we obtain the first reusable garbled circuits whose size is $|C| + \text{poly}(\lambda, d)$. For large and shallow circuits, such as those that arise from database lookup, search and some machine learning applications, this gives significant bandwidth savings over previous methods (even in the single use setting).

Theorem 1.3 (Informal). *Assuming subexponential LWE, there is a reusable circuit garbling scheme that garbles a depth- d circuit C into a circuit \hat{C} such that $|\hat{C}| = |C| + \text{poly}(\lambda, d)$, and garbles an input x into an encoded input \hat{x} such that $|\hat{x}| = |x| \cdot \text{poly}(\lambda, d)$.*

We next ask if we can obtain short garbled inputs of size $|\hat{\mathbf{x}}| = |\mathbf{x}| + \text{poly}(\lambda, d)$, analogous to what we achieved for the garbled circuit. In a beautiful recent work, Applebaum, Ishai, Kushilevitz and Waters [AIKW13] showed constructions of *single-use* garbled circuits with short garbled inputs of size $|\hat{\mathbf{x}}| = |\mathbf{x}| + \text{poly}(\lambda)$. We remark that while their garbled inputs are short, their garbled circuits still incur a multiplicative $\text{poly}(\lambda)$ overhead.

Using our second ABE scheme (based on multilinear maps) in conjunction with the techniques of Goldwasser et al. [GKP⁺13b], we obtain the first reusable garbling scheme with garbled inputs of size $|\hat{\mathbf{x}}| = |\mathbf{x}| + \text{poly}(\lambda, d)$.

Theorem 1.4 (Informal). *Assuming subexponential LWE and the existence of d -level multilinear maps, there is a reusable circuit garbling scheme that garbles a depth- d circuit C into a circuit \hat{C} such that $|\hat{C}| = |C| \cdot \text{poly}(\lambda, d)$, and garbles an input \mathbf{x} into an encoded input $\hat{\mathbf{x}}$ such that $|\hat{\mathbf{x}}| = |\mathbf{x}| + \text{poly}(\lambda, d)$.*

A natural open question is to construct a scheme which produces both short garbled circuits and short garbled inputs. We first focus on describing the ABE schemes and then give details of the garbling scheme.

ABE for arithmetic circuits. For a prime q , our first ABE system (based on LWE) directly handles arithmetic circuits with weighted addition and multiplication gates over \mathbb{Z}_q , namely gates of the form

$$g_+(x_1, \dots, x_k) = \alpha_1 x_1 + \dots + \alpha_k x_k \quad \text{and} \quad g_\times(x_1, \dots, x_k) = \alpha \cdot x_1 \cdots x_k$$

where the weights α_i can be arbitrary elements in \mathbb{Z}_q . Previous ABE constructions worked with Boolean circuits.

Addition gates g_+ take arbitrary inputs $x_1, \dots, x_k \in \mathbb{Z}_q$. However, for multiplication gates g_\times , we require that the inputs are somewhat smaller than q , namely in the range $[-p, p]$ for some $p < q$. (In fact, our construction allows for one of the inputs to g_\times to be arbitrarily large in \mathbb{Z}_q). Hence, while $f : \mathbb{Z}_q^\ell \rightarrow \mathbb{Z}_q$ can be an arbitrary polynomial-size arithmetic circuit, decryption will succeed only for attribute vectors \mathbf{x} for which $f(\mathbf{x}) = 0$ and the inputs to all multiplication gates in the circuit are in $[-p, p]$. We discuss the relation between p and q at the end of the section.

We can in turn apply our arithmetic ABE construction to Boolean circuits with large fan-in resulting in potentially large savings over constructions restricted to fan-in two gates. An AND gate can be implemented as $\wedge(x_1, \dots, x_k) = x_1 \cdots x_k$ and an OR gate as $\vee(x_1, \dots, x_k) = 1 - (1 - x_1) \cdots (1 - x_k)$. In this setting, the inputs to the gates g_+ and g_\times are naturally small, namely in $\{0, 1\}$. Thus, unbounded fan-in allows us to consider circuits with smaller size and depth, and results in smaller overall parameters.

ABE with key delegation. Our first ABE system also supports key delegation. That is, using the master secret key, user Alice can be given a secret key sk_f for a function f that lets her decrypt whenever the attribute vector \mathbf{x} satisfies $f(\mathbf{x}) = 0$. In our system, for any function g , Alice can then issue a delegated secret key $\text{sk}_{f \wedge g}$ to Bob that lets Bob decrypt if and only if the attribute vector \mathbf{x} satisfies $f(\mathbf{x}) = g(\mathbf{x}) = 0$. Bob can further delegate to Charlie, and so on. The size of the secret key increases quadratically with the number of delegations.

We note that Gorbunov et al. [GVW13] showed that their ABE system for Boolean circuits supports a somewhat restricted form of delegation. Specifically, they demonstrated that using a secret key sk_f for a function f , and a secret key sk_g for a function g , it is possible to issue a secret key $\text{sk}_{f \wedge g}$ for the function $f \wedge g$. In this light, our work resolves the naturally arising open problem of providing full delegation capabilities (i.e., issuing $\text{sk}_{f \wedge g}$ using only sk_f).

1.1 Building an ABE for arithmetic circuits with short keys

Key-homomorphic public-key encryption. We obtain our ABE by constructing a public-key encryption scheme that supports computations on public keys. Basic public keys in our system are vectors \mathbf{x} in \mathbb{Z}_q^ℓ for some ℓ . Now, let \mathbf{x} be a tuple in \mathbb{Z}_q^ℓ and let $f : \mathbb{Z}_q^\ell \rightarrow \mathbb{Z}_q$ be a function represented as a polynomial-size arithmetic circuit. Key-homomorphism means that:

anyone can transform an encryption under key \mathbf{x} into an encryption under key $f(\mathbf{x})$.

More precisely, suppose \mathbf{c} is an encryption of message μ under public-key $\mathbf{x} \in \mathbb{Z}_q^\ell$. There is a public algorithm $\text{Eval}_{\text{ct}}(f, \mathbf{x}, \mathbf{c}) \rightarrow \mathbf{c}_f$ that outputs a ciphertext \mathbf{c}_f that is an encryption of μ under the public-key $f(\mathbf{x}) \in \mathbb{Z}_q$. In our constructions Eval_{ct} is deterministic and its running time is proportional to the size of the arithmetic circuit for f .

If we give user Alice the secret-key for the public-key $0 \in \mathbb{Z}_q$ then Alice can use Eval_{ct} to decrypt \mathbf{c} whenever $f(\mathbf{x}) = 0$, as required for ABE. Unfortunately, this ABE is completely insecure! This is because the secret key is not bound to the function f : Alice could decrypt any ciphertext encrypted under \mathbf{x} by simply finding some function g such that $g(\mathbf{x}) = 0$.

To construct a secure ABE we slightly extend the basic key-homomorphism idea. A base encryption public-key is a tuple $\mathbf{x} \in \mathbb{Z}_q^\ell$ as before, however Eval_{ct} produces ciphertexts encrypted under the public key $(f(\mathbf{x}), \langle f \rangle)$ where $f(\mathbf{x}) \in \mathbb{Z}_q$ and $\langle f \rangle$ is an encoding of the circuit computing f . Transforming a ciphertext \mathbf{c} from the public key \mathbf{x} to $(f(\mathbf{x}), \langle f \rangle)$ is done using algorithm $\text{Eval}_{\text{ct}}(f, \mathbf{x}, \mathbf{c}) \rightarrow \mathbf{c}_f$ as before. To simplify the notation we write a public-key $(y, \langle f \rangle)$ as simply (y, f) . The precise syntax and security requirements for key-homomorphic public-key encryption are provided in Section 3.

To build an ABE we simply publish the parameters of the key-homomorphic PKE system. A message μ is encrypted with attribute vector $\mathbf{x} = (x_1, \dots, x_\ell) \in \mathbb{Z}_q^\ell$ that serves as the public key. Let \mathbf{c} be the resulting ciphertext. Given an arithmetic circuit f , the key-homomorphic property lets anyone transform \mathbf{c} into an encryption of μ under key $(f(\mathbf{x}), f)$. The point is that now the secret key for the function f can simply be the decryption key for the public-key $(0, f)$. This key enables the decryption of \mathbf{c} when $f(\mathbf{x}) = 0$ as follows: the decryptor first uses $\text{Eval}_{\text{ct}}(f, \mathbf{x}, \mathbf{c}) \rightarrow \mathbf{c}_f$ to transform the ciphertext to the public key $(f(\mathbf{x}), f)$. It can then decrypt \mathbf{c}_f using the decryption key it was given whenever $f(\mathbf{x}) = 0$. We show that this results in a secure ABE.

A construction from learning with errors. Fix some $n \in \mathbb{Z}^+$, prime q , and $m = \Theta(n \log q)$. Let \mathbf{A} , \mathbf{G} and $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ be matrices in $\mathbb{Z}_q^{n \times m}$ that will be part of the system parameters. To encrypt a message μ under the public key $\mathbf{x} = (x_1, \dots, x_\ell) \in \mathbb{Z}_q^\ell$ we use a variant of dual Regev encryption [Reg05, GPV08] using the following matrix as the public key:

$$(\mathbf{A} \mid x_1 \mathbf{G} + \mathbf{B}_1 \mid \dots \mid x_\ell \mathbf{G} + \mathbf{B}_\ell) \in \mathbb{Z}_q^{n \times (\ell+1)m} \quad (1)$$

We obtain a ciphertext $\mathbf{c}_\mathbf{x}$. We note that this encryption algorithm is the same as encryption in the hierarchical IBE system of [ABB10] and encryption in the predicate encryption for inner-products of [AFV11].

We show that, remarkably, this system is key-homomorphic: given a function $f : \mathbb{Z}_q^\ell \rightarrow \mathbb{Z}_q$ computed by a poly-size arithmetic circuit, anyone can transform the ciphertext $\mathbf{c}_\mathbf{x}$ into a dual Regev encryption for the public-key matrix

$$(\mathbf{A} \mid f(\mathbf{x}) \cdot \mathbf{G} + \mathbf{B}_f) \in \mathbb{Z}_q^{n \times 2m}$$

where the matrix $\mathbf{B}_f \in \mathbb{Z}_q^{n \times m}$ serves as the encoding of the circuit for the function f . This \mathbf{B}_f is uniquely determined by f and $\mathbf{B}_1, \dots, \mathbf{B}_\ell$. The work needed to compute \mathbf{B}_f is proportional to the size of the arithmetic circuit for f .

To illustrate the idea, assume that we have the ciphertext under the public key (x, y) : $\mathbf{c}_\mathbf{x} = (\mathbf{c}_0 \mid \mathbf{c}_x \mid \mathbf{c}_y)$. Here $\mathbf{c}_0 = \mathbf{A}^T \mathbf{s} + \mathbf{e}$, $\mathbf{c}_x = (x\mathbf{G} + \mathbf{B}_1)^T \mathbf{s} + \mathbf{e}_1$ and $\mathbf{c}_y = (y\mathbf{G} + \mathbf{B}_2)^T \mathbf{s} + \mathbf{e}_2$. To compute the ciphertext under the public key $(x + y, \mathbf{B}_+)$ one takes the sum of the ciphertexts \mathbf{c}_x and \mathbf{c}_y .

The result is the encryption under the matrix

$$(x + y)\mathbf{G} + (\mathbf{B}_1 + \mathbf{B}_2) \in \mathbb{Z}_q^{n \times m}$$

where $\mathbf{B}_+ = \mathbf{B}_1 + \mathbf{B}_2$. One of the main contributions of this work is a novel method of multiplying the public keys. Together with addition, described above, this gives full key-homomorphism. To construct the ciphertext under the public key (xy, \mathbf{B}_\times) , we first compute a small-norm matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$, s.t. $\mathbf{GR} = -\mathbf{B}_1$. With this in mind we compute

$$\begin{aligned} \mathbf{R}^T \mathbf{c}_y &= \mathbf{R}^T \cdot [(y\mathbf{G} + \mathbf{B}_2)^T \mathbf{s} + \mathbf{e}_2] \approx (-y\mathbf{B}_1 + \mathbf{B}_2\mathbf{R})^T \mathbf{s}, \quad \text{and} \\ y \cdot \mathbf{c}_x &= y [(x\mathbf{G} + \mathbf{B}_1)^T \mathbf{s} + \mathbf{e}_1] \approx (xy\mathbf{G} + y\mathbf{B}_1)^T \mathbf{s} \end{aligned}$$

Adding the two expressions above gives us

$$(xy\mathbf{G} + \mathbf{B}_2\mathbf{R})^T \mathbf{s} + \text{noise}$$

which is a ciphertext under the public key (xy, \mathbf{B}_\times) where $\mathbf{B}_\times = \mathbf{B}_2\mathbf{R}$. Note that performing this operation requires that we know y . This is the reason why this method gives an ABE and not (private index) predicate encryption. In Section 4.1 we show how to generalize this mechanism to arithmetic circuits with arbitrary fan-in gates.

As explained above, this key-homomorphism gives us an ABE for arithmetic circuits: the public parameters contain random matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell \in \mathbb{Z}_q^{n \times m}$ and encryption to an attribute vector \mathbf{x} in \mathbb{Z}_q^ℓ is done using dual Regev encryption to the matrix (1). A decryption key \mathbf{sk}_f for an arithmetic circuit $f : \mathbb{Z}_q^\ell \rightarrow \mathbb{Z}_q$ is a decryption key for the public-key matrix $(\mathbf{A} \mid 0 \cdot \mathbf{G} + \mathbf{B}_f) = (\mathbf{A} \mid \mathbf{B}_f)$. This key enables decryption whenever $f(\mathbf{x}) = 0$. The key \mathbf{sk}_f can be easily generated using a short basis for the lattice $\Lambda_q^\perp(\mathbf{A})$ which serves as the master secret key.

We prove selective security from the learning with errors problem (LWE) by using another homomorphic property of the system implemented in an algorithm called Eval_{sim} . Using Eval_{sim} the simulator responds to the adversary's private key queries and then solves the given LWE challenge.

Parameters and performance. Applying algorithm $\text{Eval}_{\text{ct}}(f, \mathbf{x}, \mathbf{c})$ to a ciphertext \mathbf{c} increases the magnitude of the noise in the ciphertext by a factor that depends on the depth of the circuit for f . A k -way addition gate (g_+) increases the norm of the noise by a factor of $O(km)$. A k -way multiplication gate (g_\times) where all (but one) of the inputs are in $[-p, p]$ increases the norm of the noise by a factor of $O(p^{k-1}m)$. Therefore, if the circuit for f has depth d , the noise in \mathbf{c} grows in the worst case by a factor of $O((p^{k-1}m)^d)$. Note that the weights α_i used in the gates g_+ and g_\times have no effect on the amount of noise added.

For decryption to work correctly the modulus q should be slightly larger than the noise in the ciphertext. Hence, we need q on the order of $\Omega(B \cdot (p^{k-1}m)^d)$ where B is the maximum magnitude of the noise added to the ciphertext during encryption. For security we rely on the hardness of the learning with errors (LWE) problem, which requires that the ratio q/B is not too large. In particular, the underlying problem is believed to be hard even when q/B is $2^{(n^\epsilon)}$ for some fixed $0 < \epsilon < 1/2$. In our settings $q/B = \Omega((p^{k-1}m)^d)$. Then to support circuits of depth $t(\lambda)$ for some polynomial $t(\cdot)$ we choose n such that $n \geq t(\lambda)^{(1/\epsilon)} \cdot (2 \log_2 n + k \log p)^{1/\epsilon}$, set $q = 2^{(n^\epsilon)}$, $m = \Theta(n \log q)$, and the LWE noise bound to $B = O(n)$. This ensures correctness of decryption and hardness of LWE since we have $\Omega((p^k m)^{t(\lambda)}) < q \leq 2^{(n^\epsilon)}$, as required. The ABE system of [GVW13] uses similar parameters due to a similar growth in noise as a function of circuit depth.

Secret key size. A decryption key in our system is a single $2m \times m$ low-norm matrix, namely the trapdoor for the matrix $(\mathbf{A}|\mathbf{B}_f)$. Since $m = \Theta(n \log q)$ and $\log_2 q$ grows linearly with the circuit depth d , the overall secret key size grows as $O(d^2)$ with the depth. In previous ABE systems for circuits [GVW13, GGH⁺13c] secret keys grew as $O(d^2 s)$ where s is the number of boolean gates or wires in the circuit.

Other related work. Predicate encryption [BW07, KSW08] provides a stronger privacy guarantee than ABE by additionally hiding the attribute vector \mathbf{x} . Predicate encryption systems for inner product functionalities can be built from bilinear maps [KSW08] and LWE [AFV11]. More recently, Garg et al. [GGH⁺13b] constructed functional encryption (which implies predicate encryption) for all polynomial-size functionalities using indistinguishability obfuscation.

The encryption algorithm in our system is similar to that in the hierarchical-IBE of Agrawal, Boneh, and Boyen [ABB10]. We show that this system is key-homomorphic for polynomial-size arithmetic circuits which gives us an ABE for such circuits. The first hint of the key homomorphic properties of the [ABB10] system was presented by Agrawal, Freeman, and Vaikuntanathan [AFV11] who showed that the system is key-homomorphic with respect to low-weight linear transformations and used this fact to construct a (private index) predicate encryption system for inner-products. To handle high-weight linear transformations [AFV11] used bit decomposition to represent the large weights as bits. This expands the ciphertext by a factor of $\log_2 q$, but adds more functionality to the system. Our ABE, when presented with a circuit containing only linear gates (i.e. only g_+ gates), also provides a predicate encryption system for inner products in the same security model as [AFV11], but can handle high-weight linear transformations directly, without bit decomposition, thereby obtaining shorter ciphertexts and public-keys.

A completely different approach to building circuit ABE was presented by Garg, Gentry, Sahai, and Waters [GGSW13] who showed that a general primitive they named *witness encryption* implies circuit ABE when combined with witness indistinguishable proofs.

2 Preliminaries

For a random variable X we denote by $x \leftarrow X$ the process of sampling a value x according to the distribution of X . Similarly, for a finite set S we denote by $x \leftarrow S$ the process of sampling a value x according to the uniform distribution over S . A non-negative function $\nu(\lambda)$ is *negligible* if for every polynomial $p(\lambda)$ it holds that $\nu(\lambda) \leq 1/p(\lambda)$ for all sufficiently large $\lambda \in \mathbb{N}$.

The *statistical distance* between two random variables X and Y over a finite domain Ω is defined as

$$\text{SD}(X, Y) = \frac{1}{2} \sum_{\omega \in \Omega} |\Pr[X = \omega] - \Pr[Y = \omega]|.$$

Two random variables X and Y are δ -close if $\text{SD}(X, Y) \leq \delta$. Two distribution ensembles $\{X_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are *statistically indistinguishable* if it holds that $\text{SD}(X_\lambda, Y_\lambda)$ is negligible in λ . Such random variables are *computationally indistinguishable* if for every probabilistic polynomial-time algorithm \mathcal{A} it holds that

$$\left| \Pr_{x \leftarrow X_\lambda} [\mathcal{A}(1^\lambda, x) = 1] - \Pr_{y \leftarrow Y_\lambda} [\mathcal{A}(1^\lambda, y) = 1] \right|$$

is negligible in λ .

2.1 Attribute-Based Encryption

An attribute-based encryption (ABE) scheme for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda\}$ is a quadruple $\Pi = (\text{Setup}, \text{Keygen}, \text{Enc}, \text{Dec})$ of probabilistic polynomial-time algorithms. **Setup** takes a unary representation of the security parameter λ and outputs public parameters mpk and a master secret key msk ; **Keygen**($\text{msk}, f \in \mathcal{F}_\lambda$) outputs a decryption key sk_f ; **Enc**($\text{mpk}, x \in \mathcal{X}_\lambda, \mu$) outputs a ciphertext \mathbf{c} , the encryption of message μ labeled with attribute vector x ; **Dec**(sk_f, \mathbf{c}) outputs a message μ or the special symbol \perp . (When clear from the context, we drop the subscript λ from $\mathcal{X}_\lambda, \mathcal{Y}_\lambda$ and \mathcal{F}_λ .)

Correctness. We require that for every circuit $f \in \mathcal{F}$, attribute vector $x \in \mathcal{X}$ where $f(x) = 0$, and message μ , it holds that $\text{Dec}(\text{sk}_f, \mathbf{c}) = \mu$ with an overwhelming probability over the choice of $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(\lambda)$, $\mathbf{c} \leftarrow \text{Enc}(\text{mpk}, x, \mu)$, and $\text{sk}_f \leftarrow \text{Keygen}(\text{msk}, f)$.

Security. For the most part, we consider the standard notion of selective security for ABE schemes [GPSW06]. Specifically, we consider adversaries that first announce a challenge attribute vector x^* , and then receive the public parameters mpk of the scheme and oracle access to a key-generation oracle $\text{KG}(\text{msk}, x^*, f)$ that returns the secret key sk_f for $f \in \mathcal{F}$ if $f(x^*) \neq 0$ and returns \perp otherwise. We require that any such efficient adversary has only a negligible probability in distinguishing between the ciphertexts of two different messages encrypted under the challenge attribute x^* . Formally, security is captured by the following definition.

Definition 2.1 (Selectively-secure ABE). An ABE scheme $\Pi = (\text{Setup}, \text{Keygen}, \text{Enc}, \text{Dec})$ for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda \rightarrow \mathcal{Y}_\lambda\}$ is *selectively secure* if for all probabilistic polynomial-time adversaries \mathcal{A} where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there is a negligible function $\nu(\lambda)$ such that

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{selABE}}(\lambda) \stackrel{\text{def}}{=} \left| \Pr \left[\text{EXP}_{\text{ABE}, \Pi, \mathcal{A}}^{(0)}(\lambda) = 1 \right] - \Pr \left[\text{EXP}_{\text{ABE}, \Pi, \mathcal{A}}^{(1)}(\lambda) = 1 \right] \right| \leq \nu(\lambda),$$

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$ the experiment $\text{EXP}_{\text{ABE}, \Pi, \mathcal{A}}^{(b)}(\lambda)$ is defined as follows:

1. $(x^*, \text{state}_1) \leftarrow \mathcal{A}_1(\lambda)$, where $x^* \in \mathcal{X}_\lambda$ // \mathcal{A} commits to challenge index x^*
2. $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(\lambda)$
3. $(\mu_0, \mu_1, \text{state}_2) \leftarrow \mathcal{A}_2^{\text{KG}(\text{msk}, x^*, \cdot)}(\text{mpk}, \text{state}_1)$ // \mathcal{A} outputs messages μ_0, μ_1
4. $\mathbf{c}^* \leftarrow \text{Enc}(\text{mpk}, x^*, \mu_b)$
5. $b' \leftarrow \mathcal{A}_3^{\text{KG}(\text{msk}, x^*, \cdot)}(\mathbf{c}^*, \text{state}_2)$ // \mathcal{A} outputs a guess b' for b
6. Output $b' \in \{0, 1\}$

where $\text{KG}(\text{msk}, x^*, f)$ returns a secret key $\text{sk}_f = \text{Keygen}(\text{msk}, f)$ if $f(x^*) \neq 0$ and \perp otherwise.

A fully secure ABE scheme is defined similarly, except that the adversary can choose the challenge attribute x^* after seeing the master public key and making polynomially many secret key queries. The following lemma, attributed to [BB11], says that any selectively secure ABE scheme is also fully secure with an exponential loss in parameters.

Lemma 2.2. For any selectively secure ABE scheme with attribute vectors of length $\ell = \ell(\lambda)$, there is a negligible function $\nu(\lambda)$ such that $\text{Adv}_{\Pi, \mathcal{A}}^{\text{fullABE}}(\lambda) \leq 2^{\ell(\lambda)} \cdot \nu(\lambda)$.

2.2 Background on Lattices

Lattices. Let q, n, m be positive integers. For a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ we let $\Lambda_q^\perp(\mathbf{A})$ denote the lattice $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{0} \text{ in } \mathbb{Z}_q\}$. More generally, for $\mathbf{u} \in \mathbb{Z}_q^n$ we let $\Lambda_q^\mathbf{u}(\mathbf{A})$ denote the coset $\{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}\mathbf{x} = \mathbf{u} \text{ in } \mathbb{Z}_q\}$.

We note the following elementary fact: if the columns of $\mathbf{T}_\mathbf{A} \in \mathbb{Z}^{m \times m}$ are a basis of the lattice $\Lambda_q^\perp(\mathbf{A})$, then they are also a basis for the lattice $\Lambda_q^\perp(x\mathbf{A})$ for any nonzero $x \in \mathbb{Z}_q$.

Learning with errors (LWE) [Reg05]. Fix integers n, m , a prime integer q and a noise distribution χ over \mathbb{Z} . The (n, m, q, χ) -LWE problem is to distinguish the following two distributions:

$$(\mathbf{A}, \mathbf{A}^\top \mathbf{s} + \mathbf{e}) \quad \text{and} \quad (\mathbf{A}, \mathbf{u})$$

where $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, $\mathbf{u} \leftarrow \mathbb{Z}_q^m$ are independently sampled. Throughout the paper we always set $m = \Theta(n \log q)$ and simply refer to the (n, q, χ) -LWE problem.

We say that a noise distribution χ is B -bounded if its support is in $[-B, B]$. For any fixed $d > 0$ and sufficiently large q , Regev [Reg05] (through a quantum reduction) and Peikert [Pei09] (through a classical reduction) show that taking χ as a certain q/n^d -bounded distribution, the (n, q, χ) -LWE problem is as hard as approximating the worst-case GapSVP to $n^{O(d)}$ factors, which is believed to be intractable. More generally, let $\chi_{\max} < q$ be the bound on the noise distribution. The difficulty of the LWE problem is measured by the ratio q/χ_{\max} . This ratio is always bigger than 1 and the smaller it is the harder the problem. The problem appears to remain hard even when $q/\chi_{\max} < 2^{n^\epsilon}$ for some fixed $\epsilon \in (0, 1/2)$.

Matrix norms. For a vector \mathbf{u} we let $\|\mathbf{u}\|$ denote its ℓ_2 norm. For a matrix $\mathbf{R} \in \mathbb{Z}^{k \times m}$, let $\tilde{\mathbf{R}}$ be the result of applying Gram-Schmidt (GS) orthogonalization to the columns of \mathbf{R} . We define three matrix norms:

- $\|\mathbf{R}\|$ denotes the ℓ_2 length of the longest column of \mathbf{R} .
- $\|\mathbf{R}\|_{\text{gs}} = \|\tilde{\mathbf{R}}\|$ where $\tilde{\mathbf{R}}$ is the GS orthogonalization of \mathbf{R} .
- $\|\mathbf{R}\|_2$ is the operator norm of \mathbf{R} defined as $\|\mathbf{R}\|_2 = \sup_{\|\mathbf{x}\|=1} \|\mathbf{R}\mathbf{x}\|$.

Note that $\|\mathbf{R}\|_{\text{gs}} \leq \|\mathbf{R}\| \leq \|\mathbf{R}\|_2 \leq \sqrt{k}\|\mathbf{R}\|$ and that $\|\mathbf{R} \cdot \mathbf{S}\|_2 \leq \|\mathbf{R}\|_2 \cdot \|\mathbf{S}\|_2$.

We will use the following algorithm, throughout our paper:

$\text{BD}(\mathbf{A}) \rightarrow \mathbf{R}$ where $m = n \lceil \log q \rceil$: a deterministic algorithm that takes in a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and outputs a matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$, where each element $a \in \mathbb{Z}_q$ that belongs to the matrix \mathbf{A} gets transformed into a column vector $\mathbf{r} \in \mathbb{Z}_q^{\lceil \log q \rceil}$, $\mathbf{r} = [a_0, \dots, a_{\lceil \log q \rceil - 1}]^T$. Here a_i is the i -th bit of the binary decomposition of a ordered from LSB to MSB.

Claim 2.3. For any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, matrix $\mathbf{R} = \text{BD}(\mathbf{A})$ has the norm $\|\mathbf{R}\|_2 \leq m$ and $\|\mathbf{R}^T\|_2 \leq m$.

Trapdoor generators. The following lemma states properties of algorithms for generating short basis of lattices.

Lemma 2.4. *Let $n, m, q > 0$ be integers with q prime. There are polynomial time algorithms with the properties below:*

- $\text{TrapGen}(1^n, 1^m, q) \longrightarrow (\mathbf{A}, \mathbf{T}_{\mathbf{A}})$ ([Ajt99, AP09, MP12]): a randomized algorithm that, when $m = \Theta(n \log q)$, outputs a full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and basis $\mathbf{T}_{\mathbf{A}} \in \mathbb{Z}^{m \times m}$ for $\Lambda_q^\perp(\mathbf{A})$ such that \mathbf{A} is $\text{negl}(n)$ -close to uniform and $\|\mathbf{T}_{\mathbf{A}}\|_{\text{GS}} = O(\sqrt{n \log q})$, with all but negligible probability in n .
- $\text{ExtendRight}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, \mathbf{B}) \longrightarrow \mathbf{T}_{(\mathbf{A}|\mathbf{B})}$ ([CHKP10]): a deterministic algorithm that given full-rank matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T}_{\mathbf{A}}$ of $\Lambda_q^\perp(\mathbf{A})$ outputs a basis $\mathbf{T}_{(\mathbf{A}|\mathbf{B})}$ of $\Lambda_q^\perp(\mathbf{A}|\mathbf{B})$ such that $\|\mathbf{T}_{\mathbf{A}}\|_{\text{GS}} = \|\mathbf{T}_{(\mathbf{A}|\mathbf{B})}\|_{\text{GS}}$.
- $\text{ExtendLeft}(\mathbf{A}, \mathbf{G}, \mathbf{T}_{\mathbf{G}}, \mathbf{S}) \longrightarrow \mathbf{T}_{\mathbf{H}}$ where $\mathbf{H} = (\mathbf{A} \mid \mathbf{G} + \mathbf{AS})$ ([ABB10]): a deterministic algorithm that given full-rank matrices $\mathbf{A}, \mathbf{G} \in \mathbb{Z}_q^{n \times m}$ and a basis $\mathbf{T}_{\mathbf{G}}$ of $\Lambda_q^\perp(\mathbf{G})$ outputs a basis $\mathbf{T}_{\mathbf{H}}$ of $\Lambda_q^\perp(\mathbf{H})$ such that $\|\mathbf{T}_{\mathbf{H}}\|_{\text{GS}} \leq \|\mathbf{T}_{\mathbf{G}}\|_{\text{GS}} \cdot (1 + \|\mathbf{S}\|_2)$.
- For $m = n \lceil \log q \rceil$ there is a fixed full-rank matrix $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ s.t. the lattice $\Lambda_q^\perp(\mathbf{G})$ has a publicly known basis $\mathbf{T}_{\mathbf{G}} \in \mathbb{Z}^{m \times m}$ with $\|\mathbf{T}_{\mathbf{G}}\|_{\text{GS}} \leq \sqrt{5}$. The matrix \mathbf{G} is such that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{G} \cdot \text{BD}(\mathbf{A}) = \mathbf{A}$.

To simplify the notation we will always assume that the matrix \mathbf{G} from part 4 of Lemma 2.4 has the same width m as the matrix \mathbf{A} output by algorithm TrapGen from part 1 of the lemma. We do so without loss of generality since \mathbf{G} can always be extended to the size of \mathbf{A} by adding zero columns on the right of \mathbf{G} .

Discrete Gaussians. Regev [Reg05] defined a natural distribution on $\Lambda_q^{\mathbf{u}}(\mathbf{A})$ called a *discrete Gaussian* parameterized by a scalar $\sigma > 0$. We use $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{u}}(\mathbf{A}))$ to denote this distribution. For a random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\sigma = \tilde{\Omega}(\sqrt{n})$, a vector \mathbf{x} sampled from $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{u}}(\mathbf{A}))$ has ℓ_2 norm less than $\sigma\sqrt{m}$ with probability at least $1 - \text{negl}(m)$.

For a matrix $\mathbf{U} = (\mathbf{u}_1 \mid \dots \mid \mathbf{u}_k) \in \mathbb{Z}_q^{n \times k}$ we let $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$ be a distribution on matrices in $\mathbb{Z}^{m \times k}$ where the i -th column is sampled from $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{u}_i}(\mathbf{A}))$ independently for $i = 1, \dots, k$. Clearly if \mathbf{R} is sampled from $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$ then $\mathbf{AR} = \mathbf{U}$ in \mathbb{Z}_q .

Lemma 2.5. *For integers $n, m, k, q, \sigma > 0$, matrices $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{U} \in \mathbb{Z}_q^{n \times k}$, if $\mathbf{R} \in \mathbb{Z}^{m \times k}$ is sampled from $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{U}}(\mathbf{A}))$ and \mathbf{S} is sampled uniformly in $\{\pm 1\}^{m \times m}$ then*

$$\|\mathbf{R}^\top\|_2 \leq \sigma\sqrt{mk} \quad , \quad \|\mathbf{R}\|_2 \leq \sigma\sqrt{mk} \quad , \quad \|\mathbf{S}\|_2 \leq 20\sqrt{m}$$

with overwhelming probability in m .

Proof. For the $\{\pm 1\}$ matrix \mathbf{S} the lemma follows from Litvak et al. [LPRTJ05] (Fact 2.4). For the matrix \mathbf{R} the lemma follow from the fact that $\|\mathbf{R}^\top\|_2 \leq \sqrt{k} \cdot \|\mathbf{R}\| < \sqrt{k}(\sigma\sqrt{m})$. \square

Solving $\mathbf{AX} = \mathbf{U}$. We review algorithms for finding a low-norm matrix $\mathbf{X} \in \mathbb{Z}^{m \times k}$ such that $\mathbf{AX} = \mathbf{U}$.

Lemma 2.6. *Let $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{T}_\mathbf{A} \in \mathbb{Z}^{m \times m}$ be a basis for $\Lambda_q^\perp(\mathbf{A})$. Let $\mathbf{U} \in \mathbb{Z}_q^{n \times k}$. There are polynomial time algorithms that output $\mathbf{X} \in \mathbb{Z}^{m \times k}$ satisfying $\mathbf{AX} = \mathbf{U}$ with the properties below:*

- **SampleD**($\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{U}, \sigma$) $\rightarrow \mathbf{X}$ ([GPV08]): *a randomized algorithm that, when $\sigma = \|\mathbf{T}_\mathbf{A}\|_{\text{GS}} \cdot \omega(\sqrt{\log m})$, outputs a random sample \mathbf{X} from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^\mathbf{U}(\mathbf{A}))$.*
- **RandBasis**($\mathbf{A}, \mathbf{T}_\mathbf{A}, \sigma$) $\rightarrow \mathbf{T}'_\mathbf{A}$ ([CHKP10]): *a randomized algorithm that, when $\sigma = \|\mathbf{T}_\mathbf{A}\|_{\text{GS}} \cdot \omega(\sqrt{\log m})$, outputs a basis $\mathbf{T}'_\mathbf{A}$ of $\Lambda_q^\perp(\mathbf{A})$ sampled from a distribution that is statistically close to $(\mathcal{D}_\sigma(\Lambda_q^\perp(\mathbf{A})))^m$. Note that $\|\mathbf{T}'_\mathbf{A}\|_{\text{GS}} < \sigma\sqrt{m}$ with all but negligible probability.*

Randomness extraction. We conclude with a variant of the left-over hash lemma from [ABB10].

Lemma 2.7. *Suppose that $m > (n+1)\log_2 q + \omega(\log n)$ and that $q > 2$ is prime. Let \mathbf{S} be an $m \times k$ matrix chosen uniformly in $\{1, -1\}^{m \times k} \bmod q$ where $k = k(n)$ is polynomial in n . Let \mathbf{A} and \mathbf{B} be matrices chosen uniformly in $\mathbb{Z}_q^{n \times m}$ and $\mathbb{Z}_q^{n \times k}$ respectively. Then, for all vectors \mathbf{e} in \mathbb{Z}_q^m , the distribution $(\mathbf{A}, \mathbf{AS}, \mathbf{S}^\top \mathbf{e})$ is statistically close to the distribution $(\mathbf{A}, \mathbf{B}, \mathbf{S}^\top \mathbf{e})$.*

Note that the lemma holds for every vector \mathbf{e} in \mathbb{Z}_q^m , including low norm vectors.

Additional algorithms Throughout the paper we will use the following algorithms:

- Lemma 2.8.**
- **SampleRight**($\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{B}, \mathbf{U}, \sigma$) : *a randomized algorithm that given full-rank matrices $\mathbf{A}, \mathbf{B} \in \mathbb{Z}_q^{n \times m}$, matrix $\mathbf{U} \in \mathbb{Z}_q^{n \times m}$, a basis $\mathbf{T}_\mathbf{A}$ of $\Lambda_q^\perp(\mathbf{A})$ and $\sigma = \|\mathbf{T}_\mathbf{A}\|_{\text{GS}} \cdot \omega(\sqrt{\log m})$, outputs a random sample $\mathbf{X} \in \mathbb{Z}_q^{2m \times m}$ from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^\mathbf{U}((\mathbf{A}|\mathbf{B})))$. This algorithm is the composition of two algorithms: **ExtendRight**($\mathbf{A}, \mathbf{T}_\mathbf{A}, \mathbf{B}$) $\rightarrow \mathbf{T}_{(\mathbf{A}|\mathbf{B})}$ and **SampleD**($(\mathbf{A}|\mathbf{B}), \mathbf{T}_{(\mathbf{A}|\mathbf{B})}, \mathbf{U}, \sigma$) $\rightarrow \mathbf{X}$.*
 - **SampleLeft**($\mathbf{A}, \mathbf{S}, y, \mathbf{U}, \sigma$) : *a randomized algorithm that given full-rank matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, matrices $\mathbf{S}, \mathbf{U} \in \mathbb{Z}_q^{n \times m}$, $y \neq 0 \in \mathbb{Z}_q$ and $\sigma = \sqrt{5} \cdot (1 + \|\mathbf{S}\|_2) \cdot \omega(\sqrt{\log m})$, outputs a random sample $\mathbf{X} \in \mathbb{Z}_q^{2m \times m}$ from a distribution that is statistically close to $\mathcal{D}_\sigma(\Lambda_q^\mathbf{U}((\mathbf{A}|y\mathbf{G} + \mathbf{AS})))$, where \mathbf{G} is the matrix from Lemma 2.4, part 4. This algorithm is the composition of two algorithms: **ExtendLeft**($\mathbf{A}, y\mathbf{G}, \mathbf{T}_\mathbf{G}, \mathbf{S}$) $\rightarrow \mathbf{T}_{(\mathbf{A}|y\mathbf{G} + \mathbf{AS})}$ and **SampleD**($(\mathbf{A}|y\mathbf{G} + \mathbf{AS}), \mathbf{T}_{(\mathbf{A}|y\mathbf{G} + \mathbf{AS})}, \mathbf{U}, \sigma$) $\rightarrow \mathbf{X}$.*

2.3 Multilinear Maps

Assume there exists a group generator \mathcal{G} that takes the security parameter 1^λ and the pairing bound k and outputs groups G_1, \dots, G_k each of large prime order $q > 2^\lambda$. Let g_i be the generator of group G_i and let $g = g_1$. In addition, the algorithm outputs a description of a set of bilinear maps:

$$\{e_{ij} : G_i \times G_j \rightarrow G_{i+j} \mid i, j \geq 1, i+j \leq k\}$$

satisfying $e_{ij}(g_i^a, g_j^b) = g_{i+j}^{ab}$ for all $a, b \in \mathbb{Z}_q$. We sometimes omit writing e_{ij} and for convince simply use e as the map descriptor.

Definition 2.9. $[(k, \ell)$ -Multilinear Diffie-Hellman Exponent Assumption] The challenger runs $\mathcal{G}(1^\lambda, k)$ to generate groups G_1, \dots, G_k , generators g_1, \dots, g_k and the map descriptions e_{ij} . Next, it picks $c_1, c_2, \dots, c_k \in \mathbb{Z}_q$ at random. The (k, ℓ) -MDHE problem is hard if no adversary can distinguish between the following two experiments with better than negligible advantage in λ :

$$(g^{c_1}, \dots, g^{c_1^\ell}, \dots, g^{c_1^{\ell+2}}, \dots, g^{c_1^{2\ell}}, g^{c_2}, \dots, g^{c_k}, \beta = g_k^{c_1^{\ell+1} \prod_{2 \leq i \leq k} c_i})$$

and

$$(g^{c_1}, \dots, g^{c_1^\ell}, \dots, g^{c_1^{\ell+2}}, \dots, g^{c_1^{2\ell}}, g^{c_2}, \dots, g^{c_k}, \beta)$$

where β is a randomly chosen element in G_k .

We note that if $k = 2$, then this corresponds exactly to the bilinear Diffie-Hellman Exponent Assumption (BDHE). Also, is easy to compute $g_k^{c_1^{\ell+1} \prod_{2 \leq i \leq k-1} c_i}$ by repeated pairing of the challenge components.

3 Fully Key-Homomorphic PKE (FKHE)

Our new ABE constructions are a direct application of fully key-homomorphic public-key encryption (FKHE), a notion that we introduce. Such systems are public-key encryption schemes that are homomorphic with respect to the public encryption key. We begin by precisely defining FKHE and then show that a key-policy ABE with short keys arises naturally from such a system.

Let $\{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ be sequences of finite sets. Let $\{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ be a sequence of sets of functions, namely $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda^\ell \rightarrow \mathcal{Y}_\lambda\}$ for some $\ell > 0$. Public keys in an FKHE scheme are pairs $(x, f) \in \mathcal{Y}_\lambda \times \mathcal{F}_\lambda$. We call x the “value” and f the associated function. All such pairs are valid public keys. We also allow tuples $\mathbf{x} \in \mathcal{X}_\lambda^\ell$ to function as public keys. To simplify the notation we often drop the subscript λ and simply refer to sets \mathcal{X} , \mathcal{Y} and \mathcal{F} .

In our constructions we set $\mathcal{X} = \mathbb{Z}_q$ for some q and let \mathcal{F} be the set of ℓ -variate functions on \mathbb{Z}_q computable by polynomial size arithmetic circuits.

Now, an FKHE scheme for the family of functions \mathcal{F} consists of five PPT algorithms:

- $\text{Setup}_{\text{FKHE}}(1^\lambda) \rightarrow (\text{mpk}_{\text{FKHE}}, \text{msk}_{\text{FKHE}})$: outputs a master secret key msk_{FKHE} and public parameters mpk_{FKHE} .
- $\text{KeyGen}_{\text{FKHE}}(\text{msk}_{\text{FKHE}}, (y, f)) \rightarrow \text{sk}_{y,f}$: outputs a decryption key for the public key $(y, f) \in \mathcal{Y} \times \mathcal{F}$.
- $\text{E}_{\text{FKHE}}(\text{mpk}_{\text{FKHE}}, \mathbf{x} \in \mathcal{X}^\ell, \mu) \rightarrow \mathbf{c}_{\mathbf{x}}$: encrypts message μ under the public key \mathbf{x} .
- Eval : a *deterministic* algorithm that implements key-homomorphism. Let \mathbf{c} be an encryption of message μ under public key $\mathbf{x} \in \mathcal{X}^\ell$. For a function $f : \mathcal{X}^\ell \rightarrow \mathcal{Y} \in \mathcal{F}$ the algorithm does:

$$\text{Eval}(f, \mathbf{x}, \mathbf{c}) \rightarrow \mathbf{c}_f$$

where if $y = f(x_1, \dots, x_\ell)$ then \mathbf{c}_f is an encryption of message μ under public-key (y, f) .

- $\text{D}_{\text{FKHE}}(\text{sk}_{y,f}, \mathbf{c})$: decrypts a ciphertext \mathbf{c} with key $\text{sk}_{y,f}$. If \mathbf{c} is an encryption of μ under public key (x, g) then decryption succeeds only when $x = y$ and f and g are identical arithmetic circuits.

Algorithm Eval captures the key-homomorphic property of the system: ciphertext \mathbf{c} encrypted with key $\mathbf{x} = (x_1, \dots, x_\ell)$ is transformed to a ciphertext \mathbf{c}_f encrypted under key $(f(x_1, \dots, x_\ell), f)$.

Correctness. The key-homomorphic property is stated formally in the following requirement: For all $(\text{mpk}_{\text{FKHE}}, \text{msk}_{\text{FKHE}})$ output by Setup, all messages μ , all $f \in \mathcal{F}$, and $\mathbf{x} = (x_1, \dots, x_\ell) \in \mathcal{X}^\ell$:

If $\mathbf{c} \leftarrow \text{E}_{\text{FKHE}}(\text{mpk}_{\text{FKHE}}, \mathbf{x} \in \mathcal{X}^\ell, \mu)$, $y = f(x_1, \dots, x_\ell)$,
 $\mathbf{c}_f = \text{Eval}(f, \mathbf{x}, \mathbf{c})$, $\text{sk} \leftarrow \text{KeyGen}_{\text{FKHE}}(\text{msk}_{\text{FKHE}}, (y, f))$

Then $\text{D}_{\text{FKHE}}(\text{sk}, \mathbf{c}_f) = \mu$.

An ABE from a FKHE. A FKHE for a family of functions $\mathcal{F} = \{f : \mathcal{X}^\ell \rightarrow \mathcal{Y}\}$ immediately gives a key-policy ABE. Attribute vectors for the ABE are ℓ -tuples over \mathcal{X} and the supported key-policies are functions in \mathcal{F} . The ABE system works as follows:

- **Setup** $(1^\lambda, \ell)$: Run $\text{Setup}_{\text{FKHE}}(1^\lambda)$ to get public parameters mpk and master secret msk . These function as the ABE public parameters and master secret.
- **Keygen** (msk, f) : Output $\text{sk}_f \leftarrow \text{KeyGen}_{\text{FKHE}}(\text{msk}_{\text{FKHE}}, (0, f))$.
 Jumping ahead, we remark that in our FKHE instantiation (in Section 4), the number of bits needed to encode the function f in sk_f depends only on the depth of the circuit computing f , not its size. Therefore, the size of sk_f depends only on the depth complexity of f .
- **Enc** $(\text{mpk}, \mathbf{x} \in \mathcal{X}^\ell, \mu)$: output (\mathbf{x}, \mathbf{c}) where $\mathbf{c} \leftarrow \text{E}_{\text{FKHE}}(\text{mpk}_{\text{FKHE}}, \mathbf{x}, \mu)$.
- **Dec** $(\text{sk}_f, (\mathbf{x}, \mathbf{c}))$: if $f(\mathbf{x}) = 0$ set $\mathbf{c}_f = \text{Eval}(f, \mathbf{x}, \mathbf{c})$ and output the decrypted answer $\text{D}_{\text{FKHE}}(\text{sk}_f, \mathbf{c}_f)$.
 Note that \mathbf{c}_f is the encryption of the plaintext under the public key $(f(\mathbf{x}), f)$. Since sk_f is the decryption key for the public key $(0, f)$, decryption will succeed whenever $f(\mathbf{x}) = 0$ as required.

The security of FKHE systems. Security for a fully key-homomorphic encryption system is defined so as to make the ABE system above secure. More precisely, we define security as follows.

Definition 3.1 (Selectively-secure FKHE). A fully key homomorphic encryption scheme $\Pi = (\text{Setup}_{\text{FKHE}}, \text{KeyGen}_{\text{FKHE}}, \text{E}_{\text{FKHE}}, \text{Eval})$ for a class of functions $\mathcal{F}_\lambda = \{f : \mathcal{X}_\lambda^{\ell(\lambda)} \rightarrow \mathcal{Y}_\lambda\}$ is *selectively secure* if for all p.p.t. adversaries \mathcal{A} where $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there is a negligible function $\nu(\lambda)$ such that

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{FKHE}}(\lambda) \stackrel{\text{def}}{=} \left| \Pr \left[\text{EXP}_{\text{FKHE}, \Pi, \mathcal{A}}^{(0)}(\lambda) = 1 \right] - \Pr \left[\text{EXP}_{\text{FKHE}, \Pi, \mathcal{A}}^{(1)}(\lambda) = 1 \right] \right| \leq \nu(\lambda),$$

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$ the experiment $\text{EXP}_{\text{FKHE}, \Pi, \mathcal{A}}^{(b)}(\lambda)$ is defined as:

1. $(\mathbf{x}^* \in \mathcal{X}_\lambda^{\ell(\lambda)}, \text{state}_1) \leftarrow \mathcal{A}_1(\lambda)$
2. $(\text{mpk}_{\text{FKHE}}, \text{msk}_{\text{FKHE}}) \leftarrow \text{Setup}_{\text{FKHE}}(\lambda)$
3. $(\mu_0, \mu_1, \text{state}_2) \leftarrow \mathcal{A}_2^{\text{KG}_{\text{KH}}(\text{msk}_{\text{FKHE}}, \mathbf{x}^*, \cdot, \cdot)}(\text{mpk}_{\text{FKHE}}, \text{state}_1)$

4. $\mathbf{c}^* \leftarrow \text{E}_{\text{FKHE}}(\text{mpk}_{\text{FKHE}}, \mathbf{x}^*, \mu_b)$
5. $b' \leftarrow \mathcal{A}_3^{\text{KG}_{\text{KH}}(\text{msk}_{\text{FKHE}}, x^*, \cdot, \cdot)}(\mathbf{c}^*, \text{state}_2)$ // \mathcal{A} outputs a guess b' for b
6. output $b' \in \{0, 1\}$

where $\text{KG}_{\text{KH}}(\text{msk}_{\text{FKHE}}, x^*, y, f)$ is an oracle that on input $f \in \mathcal{F}$ and $y \in \mathcal{Y}_\lambda$, returns \perp whenever $f(\mathbf{x}^*) = y$, and otherwise returns $\text{KeyGen}_{\text{FKHE}}(\text{msk}_{\text{FKHE}}, (y, f))$.

With Definition 3.1 the following theorem is now immediate.

Theorem 3.2. *The ABE system above is selectively secure provided the underlying FKHE is selectively secure.*

4 An ABE and FKHE for arithmetic circuits from LWE

We now turn to building an FKHE for arithmetic circuits from the learning with errors (LWE) problem. This directly gives an ABE with short private keys as explained in Section 3. Our construction follows the key-homomorphism paradigm outlined in the introduction.

For integers n and $q = q(n)$ let $m = \Theta(n \log q)$. Let $\mathbf{G} \in \mathbb{Z}_q^{n \times m}$ be the fixed matrix from Lemma 2.4 (part 4). For $x \in \mathbb{Z}_q$, $\mathbf{B} \in \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \in \mathbb{Z}_q^n$, and $\delta > 0$ define the set

$$E_{\mathbf{s}, \delta}(x, \mathbf{B}) = \{(x\mathbf{G} + \mathbf{B})^\top \mathbf{s} + \mathbf{e} \in \mathbb{Z}_q^m \text{ where } \|\mathbf{e}\| < \delta\}$$

For now we will assume the existence of three efficient *deterministic* algorithms Eval_{pk} , Eval_{ct} , Eval_{sim} that implement the key-homomorphic features of the scheme and are at the heart of the construction. We present them in the next section. These three algorithms must satisfy the following properties with respect to some family of functions $\mathcal{F} = \{f : (\mathbb{Z}_q)^\ell \rightarrow \mathbb{Z}_q\}$ and a function $\alpha_{\mathcal{F}} : \mathbb{Z} \rightarrow \mathbb{Z}$.

- $\text{Eval}_{\text{pk}}(f \in \mathcal{F}, \vec{\mathbf{B}} \in (\mathbb{Z}_q^{n \times m})^\ell) \rightarrow \mathbf{B}_f \in \mathbb{Z}_q^{n \times m}$.
- $\text{Eval}_{\text{ct}}(f \in \mathcal{F}, ((x_i, \mathbf{B}_i, \mathbf{c}_i))_{i=1}^\ell) \rightarrow \mathbf{c}_f \in \mathbb{Z}_q^m$. Here $x_i \in \mathbb{Z}_q$, $\mathbf{B}_i \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{c}_i \in E_{\mathbf{s}, \delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$. Note that the same \mathbf{s} is used for all \mathbf{c}_i . The output \mathbf{c}_f must satisfy

$$\mathbf{c}_f \in E_{\mathbf{s}, \Delta}(f(\mathbf{x}), \mathbf{B}_f) \text{ where } \mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$$

and $\mathbf{x} = (x_1, \dots, x_\ell)$. We further require that $\Delta < \delta \cdot \alpha_{\mathcal{F}}(n)$ for some function $\alpha_{\mathcal{F}}(n)$ that measures the increase in the noise magnitude in \mathbf{c}_f compared to the input ciphertexts.

This algorithm captures the key-homomorphic property: it translates ciphertexts encrypted under public-keys $\{x_i\}_{i=1}^\ell$ into a ciphertext \mathbf{c}_f encrypted under public-key $(f(\mathbf{x}), f)$.

- $\text{Eval}_{\text{sim}}(f \in \mathcal{F}, ((x_i^*, \mathbf{S}_i))_{i=1}^\ell, \mathbf{A}) \rightarrow \mathbf{S}_f \in \mathbb{Z}_q^{m \times m}$. Here $x_i^* \in \mathbb{Z}_q$ and $\mathbf{S}_i \in \mathbb{Z}_q^{m \times m}$. With $\mathbf{x}^* = (x_1^*, \dots, x_n^*)$, the output \mathbf{S}_f satisfies

$$\mathbf{A}\mathbf{S}_f - f(\mathbf{x}^*)\mathbf{G} = \mathbf{B}_f \text{ where } \mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{A}\mathbf{S}_1 - x_1^*\mathbf{G}, \dots, \mathbf{A}\mathbf{S}_\ell - x_\ell^*\mathbf{G})) .$$

We further require that for all $f \in \mathcal{F}$, if $\mathbf{S}_1, \dots, \mathbf{S}_\ell$ are random matrices in $\{\pm 1\}^{m \times m}$ then $\|\mathbf{S}_f\|_2 < \alpha_{\mathcal{F}}(n)$ with all but negligible probability.

Definition 4.1. The deterministic algorithms $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ are $\alpha_{\mathcal{F}}$ -FKHE enabling for some family of functions $\mathcal{F} = \{f : (\mathbb{Z}_q)^\ell \rightarrow \mathbb{Z}_q\}$ if there are functions $q = q(n)$ and $\alpha_{\mathcal{F}} = \alpha_{\mathcal{F}}(n)$ for which the properties above are satisfied.

We want $\alpha_{\mathcal{F}}$ -FKHE enabling algorithms for a large function family \mathcal{F} and the smallest possible $\alpha_{\mathcal{F}}$. In the next section we build these algorithms for polynomial-size arithmetic circuits. The function $\alpha_{\mathcal{F}}(n)$ will depend on the depth of circuits in the family.

The FKHE system. Given FKHE-enabling algorithms $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ for a family of functions $\mathcal{F} = \{f : (\mathbb{Z}_q)^\ell \rightarrow \mathbb{Z}_q\}$ we build an FKHE for the same family of functions \mathcal{F} . We prove selective security based on the learning with errors problem.

- **Parameters :** Choose n and $q = q(n)$ as needed for $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ to be $\alpha_{\mathcal{F}}$ -FKHE enabling for the function family \mathcal{F} . In addition, let χ be a χ_{\max} -bounded noise distribution for which the (n, q, χ) -LWE problem is hard as discussed in Appendix 2.2. As usual, we set $m = \Theta(n \log q)$.

Set $\sigma = \omega(\alpha_{\mathcal{F}} \cdot \sqrt{\log m})$. We instantiate these parameters concretely in the next section.

For correctness of the scheme we require that $\alpha_{\mathcal{F}}^2 \cdot m < \frac{1}{12} \cdot (q/\chi_{\max})$ and $\alpha_{\mathcal{F}} > \sqrt{n \log m}$.

- **Setup_{FKHE}** $(1^\lambda) \rightarrow (\text{mpk}_{\text{FKHE}}, \text{msk}_{\text{FKHE}})$: Run algorithm $\text{TrapGen}(1^n, 1^m, q)$ from Lemma 2.4 (part 1) to generate $(\mathbf{A}, \mathbf{T}_{\mathbf{A}})$ where \mathbf{A} is a uniform full-rank matrix in $\mathbb{Z}_q^{n \times m}$. Choose random matrices $\mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell \in \mathbb{Z}_q^{n \times m}$ and output a master secret key msk_{FKHE} and public parameters mpk_{FKHE} :

$$\text{mpk}_{\text{FKHE}} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell) \quad ; \quad \text{msk}_{\text{FKHE}} = (\mathbf{T}_{\mathbf{A}})$$

- **KeyGen_{FKHE}** $(\text{msk}_{\text{FKHE}}, (y, f)) \rightarrow \text{sk}_{y,f}$: Let $\mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$. Output $\text{sk}_{y,f} := \mathbf{R}_f$ where \mathbf{R}_f is a low-norm matrix in $\mathbb{Z}^{2m \times m}$ sampled from the discrete Gaussian distribution $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{D}}(\mathbf{A}|y\mathbf{G} + \mathbf{B}_f))$ so that $(\mathbf{A}|y\mathbf{G} + \mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$.

To construct \mathbf{R}_f run algorithm $\text{SampleRight}(\mathbf{A}, \mathbf{T}_{\mathbf{A}}, y\mathbf{G} + \mathbf{B}_f, \mathbf{D}, \sigma)$ from Lemma 2.8, part 1. Here σ is sufficiently large for algorithm SampleRight since $\sigma = \|\mathbf{T}_{\mathbf{A}}\|_{\text{gs}} \cdot \omega(\sqrt{\log m})$, where $\|\mathbf{T}_{\mathbf{A}}\|_{\text{gs}} = O(\sqrt{n \log q})$.

Note that the secret key $\text{sk}_{y,f}$ is always in $\mathbb{Z}^{2m \times m}$ independent of the complexity of the function f . We assume $\text{sk}_{y,f}$ also implicitly includes mpk_{FKHE} .

- **E_{FKHE}** $(\text{mpk}_{\text{FKHE}}, \mathbf{x} \in \mathcal{X}^\ell, \mu) \rightarrow \mathbf{c}_{\mathbf{x}}$: Choose a random n dimensional vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and error vectors $\mathbf{e}_0, \mathbf{e}_1 \leftarrow \chi^m$. Choose ℓ uniformly random matrices $\mathbf{S}_i \leftarrow \{\pm 1\}^{m \times m}$ for $i \in [\ell]$. Set $\mathbf{H} \in \mathbb{Z}_q^{n \times (\ell+1)m}$ and $\mathbf{e} \in \mathbb{Z}_q^{(\ell+1)m}$ as

$$\begin{aligned} \mathbf{H} &= (\mathbf{A} \mid x_1\mathbf{G} + \mathbf{B}_1 \mid \dots \mid x_\ell\mathbf{G} + \mathbf{B}_\ell) \in \mathbb{Z}_q^{n \times (\ell+1)m} \\ \mathbf{e} &= (\mathbf{I}_m \mid \mathbf{S}_1 \mid \dots \mid \mathbf{S}_\ell)^\top \cdot \mathbf{e}_0 \in \mathbb{Z}_q^{(\ell+1)m} \end{aligned}$$

Let $\mathbf{c}_{\mathbf{x}} = (\mathbf{H}^T \mathbf{s} + \mathbf{e}, \mathbf{D}^T \mathbf{s} + \mathbf{e}_1 + \lceil q/2 \rceil \mu) \in \mathbb{Z}_q^{(\ell+2)m}$. Output the ciphertext $\mathbf{c}_{\mathbf{x}}$.

- $D_{\text{FKHE}}(\text{sk}_{y,f}, \mathbf{c})$: Let \mathbf{c} be the encryption of μ under public key (x, g) . If $x \neq y$ or f and g are not identical arithmetic circuits, output \perp . Otherwise, let $\mathbf{c} = (\mathbf{c}_{in}, \mathbf{c}_1, \dots, \mathbf{c}_\ell, \mathbf{c}_{out}) \in \mathbb{Z}_q^{(\ell+2)m}$.

Set $\mathbf{c}_f = \text{Eval}_{\text{ct}}(f, \{(x_i, \mathbf{B}_i, \mathbf{c}_i)\}_{i=1}^\ell) \in \mathbb{Z}_q^m$.

Let $\mathbf{c}'_f = (\mathbf{c}_{in} | \mathbf{c}_f) \in \mathbb{Z}_q^{2m}$ and output $\text{Round}(\mathbf{c}_{out} - \mathbf{R}_f^\top \mathbf{c}'_f) \in \{0, 1\}^m$.

This completes the description of the system.

Correctness. The correctness of the scheme follows from our choice of parameters and, in particular, from the requirement $\alpha_{\mathcal{F}}^2 \cdot m < \frac{1}{12} \cdot (q/\chi_{\max})$. Specifically, to show correctness, first note that when $f(\mathbf{x}) = y$ we know by the requirement on Eval_{ct} that \mathbf{c}_f is in $E_{\mathbf{s}, \Delta}(y, \mathbf{B}_f)$ so that $\mathbf{c}_f = y\mathbf{G} + \mathbf{B}_f^\top \mathbf{s} + \mathbf{e}$ with $\|\mathbf{e}\| < \Delta$. Consequently,

$$\mathbf{c}'_f = (\mathbf{c}_{in} | \mathbf{c}_f) = (\mathbf{A} | y\mathbf{G} + \mathbf{B}_f)^\top \mathbf{s} + \mathbf{e}' \quad \text{where} \quad \|\mathbf{e}'\| < \Delta + \chi_{\max} < (\alpha_{\mathcal{F}} + 1)\chi_{\max}.$$

Since $\mathbf{R}_f \in \mathbb{Z}^{2m \times m}$ is sampled from the distribution $\mathcal{D}_\sigma(\Lambda_q^{\mathbf{D}}(\mathbf{A} | y\mathbf{G} + \mathbf{B}_f))$ we know that $(\mathbf{A} | y\mathbf{G} + \mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$ and, by Lemma 2.5, $\|\mathbf{R}_f^\top\|_2 < 2m\sigma$ with overwhelming probability. Therefore

$$\mathbf{c}_{out} - \mathbf{R}_f^\top \mathbf{c}'_f = (\mathbf{D}^\top \mathbf{s} + \mathbf{e}_1) - (\mathbf{D}^\top \mathbf{s} + \mathbf{R}_f^\top \mathbf{e}') = \mathbf{e}_1 - \mathbf{R}_f^\top \mathbf{e}'$$

and $\|\mathbf{e}_1 - \mathbf{R}_f^\top \mathbf{e}'\| \leq \chi_{\max} + 2m\sigma \cdot (\alpha_{\mathcal{F}} + 1)\chi_{\max} \leq 3\alpha_{\mathcal{F}}^2 \cdot \chi_{\max} \cdot m$ with overwhelming probability. By the bounds on $\alpha_{\mathcal{F}}$ this quantity is less than $q/4$ thereby ensuring correct decryption of all bits of $\mu \in \{0, 1\}^m$.

Security. Next we prove that our FKHE is selectively secure for the family of functions \mathcal{F} for which algorithms $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ are *FKHE-enabling*.

Theorem 4.2. *Given the three algorithms $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ for the family of functions \mathcal{F} , the FKHE system above is selectively secure with respect to \mathcal{F} , assuming the (n, q, χ) -LWE assumption holds where n, q, χ are the parameters for the FKHE.*

Proof idea. Before giving the complete proof we first briefly sketch the main proof idea which hinges on the properties of algorithms $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ and also employs ideas from [CHKP10, ABB10]. We build an LWE algorithm \mathcal{B} that uses a selective FKHE attacker \mathcal{A} to solve LWE. \mathcal{B} is given an LWE challenge matrix $(\mathbf{A} | \mathbf{D}) \in \mathbb{Z}_q^{n \times 2m}$ and two vectors $\mathbf{c}_{in}, \mathbf{c}_{out} \in \mathbb{Z}_q^m$ that are either random or their concatenation equals $(\mathbf{A} | \mathbf{D})^\top \mathbf{s} + \mathbf{e}$ for some small noise vector \mathbf{e} .

\mathcal{A} starts by committing to the target attribute vector $\mathbf{x} = (x_1^*, \dots, x_\ell^*) \in \mathbb{Z}_q^\ell$. In response \mathcal{B} constructs the FKHE public parameters by choosing random matrices $\mathbf{S}_1^*, \dots, \mathbf{S}_\ell^*$ in $\{\pm 1\}^{m \times m}$ and setting $\mathbf{B}_i = \mathbf{A} \mathbf{S}_i^* - x_i^* \mathbf{G}$. It gives \mathcal{A} the public parameters $\text{mpk}_{\text{FKHE}} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell)$. A standard argument shows that each of $\mathbf{A} \mathbf{S}_i^*$ is uniformly distributed in $\mathbb{Z}_q^{n \times m}$ so that all \mathbf{B}_i are uniform as required for the public parameters.

Now, consider a private key query from \mathcal{A} for a function $f \in \mathcal{F}$ and attribute $y \in \mathbb{Z}_q$. Only functions f and attributes y for which $y^* = f(x_1^*, \dots, x_\ell^*) \neq y$ are allowed. Let $\mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$. Then \mathcal{B} needs to produce a matrix \mathbf{R}_f in $\mathbb{Z}^{2m \times m}$ satisfying $(\mathbf{A} | \mathbf{B}_f) \cdot \mathbf{R}_f =$

D. To do so \mathcal{B} needs a recoding matrix from the lattice $\Lambda_q^\perp(\mathbf{F})$ where $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f)$ to the lattice $\Lambda_q^\perp(\mathbf{D})$. In the real key generation algorithm this short basis is derived from a short basis for $\Lambda_q^\perp(\mathbf{A})$ using algorithm `SampleRight`. Unfortunately, \mathcal{B} has no short basis for $\Lambda_q^\perp(\mathbf{A})$.

Instead, as explained below, \mathcal{B} builds a low-norm matrix $\mathbf{S}_f \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{B}_f = \mathbf{A}\mathbf{S}_f - y^*\mathbf{G}$. Because $y^* \neq y$, algorithm \mathcal{B} can construct the required key as $\mathbf{R}_f \leftarrow \text{SampleLeft}(\mathbf{A}, \mathbf{S}_f, (y - y^*), \mathbf{D}, \sigma)$.

The remaining question is how does algorithm \mathcal{B} build a low-norm matrix $\mathbf{S}_f \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{B}_f = \mathbf{A}\mathbf{S}_f - y^*\mathbf{G}$. To do so \mathcal{B} uses `Evalsim` giving it the secret matrices \mathbf{S}_i^* . More precisely, \mathcal{B} runs `Evalsim`($f, ((x_i^*, \mathbf{S}_i^*))_{i=1}^\ell, \mathbf{A}$) and obtains the required \mathbf{S}_f . This lets \mathcal{B} answer all private key queries.

To complete the proof it is not difficult to show that \mathcal{B} can build a challenge ciphertext \mathbf{c}^* for the attribute vector $\mathbf{x} \in \mathbb{Z}_q^\ell$ that lets it solve the given LWE instance using adversary \mathcal{A} . An important point is that \mathcal{B} cannot construct a key that decrypts \mathbf{c}^* . The reason is that it cannot build a secret key $\text{sk}_{y,f}$ for functions where $f(\mathbf{x}^*) = y$ and these are the only keys that will decrypt \mathbf{c}^* .

Proof of Theorem 4.2. The proof proceeds in a sequence of games where the first game is identical to the ABE game from Definition 2.1. In the last game in the sequence the adversary has advantage zero. We show that a PPT adversary cannot distinguish between the games which will prove that the adversary has negligible advantage in winning the original ABE security game. The LWE problem is used in proving that Games 2 and 3 are indistinguishable.

Game 0. This is the original ABE security game from Definition 2.1 between an attacker \mathcal{A} against our scheme and an ABE challenger.

Game 1. Recall that in Game 0 part of the public parameters mpk are generated by choosing random matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ in $\mathbb{Z}_q^{n \times m}$. At the challenge phase (step 4 in Definition 2.1) a challenge ciphertext \mathbf{c}^* is generated. We let $\mathbf{S}_1^*, \dots, \mathbf{S}_\ell^* \in \{-1, 1\}^{m \times m}$ denote the random matrices generated for the creation of \mathbf{c}^* in the encryption algorithm `Enc`.

In Game 1 we slightly change how the matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ are generated for the public parameters. Let $\mathbf{x}^* = (x_1^*, \dots, x_\ell^*) \in \mathbb{Z}_q^\ell$ be the target point that \mathcal{A} intends to attack. In Game 1 the random matrices $\mathbf{S}_1^*, \dots, \mathbf{S}_\ell^*$ in $\{\pm 1\}^{m \times m}$ are chosen at the setup phase (step 2) and the matrices $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ are constructed as

$$\mathbf{B}_i := \mathbf{A}\mathbf{S}_i^* - x_i^*\mathbf{G} \quad (2)$$

The remainder of the game is unchanged.

We show that Game 0 is statistically indistinguishable from Game 1 by Lemma 2.7. Observe that in Game 1 the matrices \mathbf{S}_i^* are used only in the construction of \mathbf{B}_i and in the construction of the challenge ciphertext where $\mathbf{e} := (\mathbf{I}_m|\mathbf{S}_1^*|\dots|\mathbf{S}_\ell^*)^\top \cdot \mathbf{e}_0$ is used as the noise vector for some $\mathbf{e}_0 \in \mathbb{Z}_q^m$. Let $\mathbf{S}^* = (\mathbf{S}_1^*|\dots|\mathbf{S}_\ell^*)$, then by Lemma 2.7 the distribution $(\mathbf{A}, \mathbf{A}\mathbf{S}^*, \mathbf{e})$ is statistically close to the distribution $(\mathbf{A}, \mathbf{A}', \mathbf{e})$ where \mathbf{A}' is a uniform matrix in $\mathbb{Z}_q^{n \times \ell m}$. It follows that in the adversary's view, all the matrices $\mathbf{A}\mathbf{S}_i^*$ are statistically close to uniform and therefore the \mathbf{B}_i as defined in (2) are close to uniform. Hence, the \mathbf{B}_i in Games 0 and 1 are statistically indistinguishable.

Game 2. We now change how \mathbf{A} in mpk is chosen. In Game 2 we generate \mathbf{A} as a random matrix in $\mathbb{Z}_q^{n \times m}$. The construction of $\mathbf{B}_1, \dots, \mathbf{B}_\ell$ remains as in Game 1, namely $\mathbf{B}_i = \mathbf{A}\mathbf{S}_i^* - x_i^*\mathbf{G}$.

The key generation oracle responds to private key queries (in steps 3 and 5 of Definition 2.1) using the trapdoor $\mathbf{T}_\mathbf{G}$. Consider a private key query for function $f \in \mathcal{F}$ and element $y \in \mathcal{Y}$. Only

f such that $y^* = f(x_1^*, \dots, x_\ell^*) \neq y$ are allowed. To respond, the key generation oracle computes $\mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$ and needs to produce a matrix \mathbf{R}_f in $\mathbb{Z}^{2m \times m}$ satisfying

$$(\mathbf{A}|y\mathbf{G} + \mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D} \quad \text{in } \mathbb{Z}_q.$$

To do so the key generation oracle does:

- It runs $\mathbf{S}_f \leftarrow \text{Eval}_{\text{sim}}(f, ((x_i^*, \mathbf{S}_i^*))_{i=1}^\ell, \mathbf{A})$ and obtains a low-norm matrix $\mathbf{S}_f \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{A}\mathbf{S}_f - y^*\mathbf{G} = \mathbf{B}_f$. By definition of Eval_{sim} we know that $\|\mathbf{S}_f\|_2 \leq \alpha_{\mathcal{F}}$.
- Finally, it responds with $\mathbf{R}_f = \text{SampleLeft}(\mathbf{A}, \mathbf{S}_f, y, \mathbf{D}, \sigma)$. By definition of SampleLeft we know that \mathbf{R}_f is distributed as required. Indeed because $\|\mathbf{S}_f\|_2 \leq \alpha_{\mathcal{F}}(n)$, $\sigma = \sqrt{5} \cdot (1 + \|\mathbf{S}_f\|_2) \cdot \omega(\sqrt{\log m})$ as needed for algorithm SampleLeft in Lemma 2.8, part 2.

Game 2 is otherwise the same as Game 1. Since the public parameters and responses to private key queries are statistically close to those in Game 1, the adversary's advantage in Game 2 is at most negligibly different from its advantage in Game 1.

Game 3. Game 3 is identical to Game 2 except that in the challenge ciphertext $(\mathbf{x}^*, \mathbf{c}^*)$ the vector $\mathbf{c}^* = (\mathbf{c}_{in}|\mathbf{c}_1|\dots|\mathbf{c}_\ell|\mathbf{c}_{out}) \in \mathbb{Z}_q^{(\ell+2)m}$ is chosen as a random independent vector in $\mathbb{Z}_q^{(\ell+2)m}$. Since the challenge ciphertext is always a fresh random element in the ciphertext space, \mathcal{A} 's advantage in this game is zero.

It remains to show that Game 2 and Game 3 are computationally indistinguishable for a PPT adversary, which we do by giving a reduction from the LWE problem.

Reduction from LWE. Suppose \mathcal{A} has non-negligible advantage in distinguishing Games 2 and 3. We use \mathcal{A} to construct an LWE algorithm \mathcal{B} .

LWE Instance. \mathcal{B} begins by obtaining an LWE challenge consisting of two random matrices \mathbf{A}, \mathbf{D} in $\mathbb{Z}_q^{n \times m}$ and two vectors $\mathbf{c}_{in}, \mathbf{c}_{out}$ in \mathbb{Z}_q^m . We know that $\mathbf{c}_{in}, \mathbf{c}_{out}$ are either random in \mathbb{Z}_q^m or

$$\mathbf{c}_{in} = \mathbf{A}^\top \mathbf{s} + \mathbf{e}_0 \quad \text{and} \quad \mathbf{c}_{out} = \mathbf{D}^\top \mathbf{s} + \mathbf{e}_1 \quad (3)$$

for some random vector $\mathbf{s} \in \mathbb{Z}_q^n$ and $\mathbf{e}_0, \mathbf{e}_1 \leftarrow \chi^m$. Algorithm \mathcal{B} 's goal is to distinguish these two cases with non-negligible advantage by using \mathcal{A} .

Public parameters. \mathcal{A} begins by committing to a target point $\mathbf{x} = (x_1^*, \dots, x_\ell^*) \in \mathbb{Z}_q^m$ where it wishes to be challenged. \mathcal{B} assembles the public parameters mpk as in Game 2: choose random matrices $\mathbf{S}_1^*, \dots, \mathbf{S}_\ell^*$ in $\{\pm 1\}^{m \times m}$ and set $\mathbf{B}_i = \mathbf{A}\mathbf{S}_i^* - x_i^*\mathbf{G}$. It gives \mathcal{A} the public parameters

$$\text{mpk} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell)$$

Private key queries. \mathcal{B} answers \mathcal{A} 's private-key queries (in steps 3 and 5 of Definition 2.1) as in Game 2.

Challenge ciphertext. When \mathcal{B} receives two messages $\mu_0, \mu_1 \in \{0, 1\}^m$ from \mathcal{A} , it prepares a challenge ciphertext by choosing a random $b \leftarrow \{0, 1\}$ and computing

$$\mathbf{c}_0^* = (\mathbf{I}_m|\mathbf{S}_1^*|\dots|\mathbf{S}_\ell^*)^\top \cdot \mathbf{c}_{in} \in \mathbb{Z}_q^{(\ell+1)m} \quad (4)$$

and $\mathbf{c}^* = (\mathbf{c}_0^*, \mathbf{c}_{out} + \lceil q/2 \rceil \mu_b) \in \mathbb{Z}_q^{(\ell+2)m}$. \mathcal{B} sends $(\mathbf{x}^*, \mathbf{c}^*)$ as the challenge ciphertext to \mathcal{A} .

We argue that when the LWE challenge is pseudorandom (namely (3) holds) then \mathbf{c}^* is distributed exactly as in Game 2. First, observe that when encrypting (\mathbf{x}^*, μ_b) the matrix \mathbf{H} constructed in the encryption algorithm Enc is

$$\begin{aligned} \mathbf{H} &= (\mathbf{A} \mid x_1^* \mathbf{G} + \mathbf{B}_1 \mid \cdots \mid x_\ell^* \mathbf{G} + \mathbf{B}_\ell) \\ &= (\mathbf{A} \mid x_1^* \mathbf{G} + (\mathbf{A}\mathbf{S}_1^* - x_1^* \mathbf{G}) \mid \cdots \mid x_\ell^* \mathbf{G} + (\mathbf{A}\mathbf{S}_\ell^* - x_\ell^* \mathbf{G})) = (\mathbf{A} \mid \mathbf{A}\mathbf{S}_1^* \mid \cdots \mid \mathbf{A}\mathbf{S}_\ell^*) \end{aligned}$$

Therefore, \mathbf{c}_0^* defined in (4) satisfies:

$$\begin{aligned} \mathbf{c}_0^* &= (\mathbf{I}_m \mid \mathbf{S}_1^* \mid \cdots \mid \mathbf{S}_\ell^*)^\top \cdot (\mathbf{A}^\top \mathbf{s} + \mathbf{e}_0) \\ &= (\mathbf{A} \mid \mathbf{A}\mathbf{S}_1^* \mid \cdots \mid \mathbf{A}\mathbf{S}_\ell^*)^\top \cdot \mathbf{s} + (\mathbf{I}_m \mid \mathbf{S}_1^* \mid \cdots \mid \mathbf{S}_\ell^*)^\top \cdot \mathbf{e}_0 = \mathbf{H}^\top \mathbf{s} + \mathbf{e} \end{aligned}$$

where $\mathbf{e} = (\mathbf{I}_m \mid \mathbf{S}_1^* \mid \cdots \mid \mathbf{S}_\ell^*)^\top \cdot \mathbf{e}_0$. This \mathbf{e} is sampled from the same distribution as the noise vector \mathbf{e} in algorithm Enc . We therefore conclude that \mathbf{c}_0^* is computed as in Game 2. Moreover, since $\mathbf{c}_{out} = \mathbf{D}^\top \mathbf{s} + \mathbf{e}_1$ we know that the entire challenge ciphertext \mathbf{c}^* is a valid encryption of (\mathbf{x}^*, μ_b) as required.

When the LWE challenge is random we know that \mathbf{c}_{in} and \mathbf{c}_{out} are uniform in \mathbb{Z}_q^m . Therefore the public parameters and \mathbf{c}_0^* defined in (4) are uniform and independent in $\mathbb{Z}_q^{(\ell+1)m}$ by a standard application of the left over hash lemma (e.g. Theorem 8.38 of [Sho08]) where the universal hash function is defined as multiplication by the random matrix $(\mathbf{A}^\top \mid \mathbf{c}_{in})^\top$. Since \mathbf{c}_{out} is also uniform, the challenge ciphertext overall is uniform in $\mathbb{Z}_q^{(\ell+2)m}$, as in Game 3.

Guess. Finally, \mathcal{A} guesses if it is interacting with a Game 2 or Game 3 challenger. \mathcal{B} outputs \mathcal{A} 's guess as the answer to the LWE challenge it is trying to solve.

We already argued that when the LWE challenge is pseudorandom the adversary's view is as in Game 2. When the LWE challenge is random the adversary's view is as in Game 3. Hence, \mathcal{B} 's advantage in solving LWE is the same as \mathcal{A} 's advantage in distinguishing Games 2 and 3, as required. This completes the description of algorithm \mathcal{B} and completes the proof. \blacksquare

Remark 4.3. We note that the matrix \mathbf{R}_f in $\text{KeyGen}_{\text{FKHE}}$ can alternatively be generated using a sampling method from [MP12]. To do so we choose FKHE public parameters as we do in the security proof by choosing random matrices $\mathbf{S}_i, \dots, \mathbf{S}_\ell$ in $\{\pm 1\}^{m \times m}$ and setting $\mathbf{B}_i = \mathbf{A} \mathbf{S}_i$. We then define the matrix \mathbf{B}_f as $\mathbf{B}_f := \mathbf{A} \mathbf{S}_f$ where $\mathbf{S}_f = \text{Eval}_{\text{sim}}(f, ((0, \mathbf{S}_i))_{i=1}^\ell, \mathbf{A})$. We could then build the secret key matrix $\mathbf{sk}_{y,f} = \mathbf{R}_f$ satisfying $(\mathbf{A} \mid y \mathbf{G} + \mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$ directly from the bit decomposition of \mathbf{D}/y . Adding suitable low-norm noise to the result will ensure that $\mathbf{sk}_{y,f}$ is distributed as in the simulation in the security proof. Note that this approach can only be used to build secret keys $\mathbf{sk}_{y,f}$ when $y \neq 0$ where as the method in $\text{KeyGen}_{\text{FKHE}}$ works for all y .

4.1 Evaluation Algorithms for Arithmetic Circuits

In this section we build the *FKHE-enabling* algorithms ($\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}}$) that are at the heart of the FKHE construction in Section 4. We do so for the family of polynomial depth, unbounded fan-in arithmetic circuits.

4.2 Evaluation algorithms for gates

We first describe **Eval** algorithms for single gates, i.e. when \mathcal{G} is the set of functions that each takes k inputs and computes either weighted addition or multiplication:

$$\mathcal{G} = \bigcup_{\alpha, \alpha_1, \dots, \alpha_k \in \mathbb{Z}_q} \left\{ g \mid g : \mathbb{Z}_q^k \rightarrow \mathbb{Z}_q, \begin{array}{l} g(x_1, \dots, x_k) = \alpha_1 x_1 + \alpha_2 x_2 + \dots + \alpha_k x_k \\ \text{or} \\ g(x_1, \dots, x_k) = \alpha \cdot x_1 \cdot x_2 \cdot \dots \cdot x_k \end{array} \right\} \quad (5)$$

We assume that all the inputs to a multiplication gate (except possibly one input) are integers in the interval $[-p, p]$ for some bound $p < q$.

We present all three deterministic **Eval** algorithms at once:

$$\begin{aligned} \text{Eval}_{\text{pk}}(g \in \mathcal{G}, \vec{\mathbf{B}} \in (\mathbb{Z}_q^{n \times m})^k) &\longrightarrow \mathbf{B}_g \in \mathbb{Z}_q^{n \times m} \\ \text{Eval}_{\text{ct}}(g \in \mathcal{G}, ((x_i, \mathbf{B}_i, \mathbf{c}_i))_{i=1}^k) &\longrightarrow \mathbf{c}_g \in \mathbb{Z}_q^m \\ \text{Eval}_{\text{sim}}(g \in \mathcal{G}, ((x_i^*, \mathbf{S}_i))_{i=1}^k, \mathbf{A}) &\longrightarrow \mathbf{S}_g \in \mathbb{Z}_q^{m \times m} \end{aligned}$$

- For a weighted **addition** gate $g(x_1, \dots, x_k) = \alpha_1 x_1 + \dots + \alpha_k x_k$ do:
For $i \in [k]$ generate matrix $\mathbf{R}_i \in \mathbb{Z}_q^{m \times m}$ such that

$$\mathbf{G}\mathbf{R}_i = \alpha_i \mathbf{G} : \mathbf{R}_i = \text{BD}(\alpha_i \mathbf{G}) \quad (\text{as in Lemma 2.4 part 4}). \quad (6)$$

Output the following matrices and the ciphertext:

$$\mathbf{B}_g = \sum_{i=1}^k \mathbf{B}_i \mathbf{R}_i, \quad \mathbf{S}_g = \sum_{i=1}^k \mathbf{S}_i \mathbf{R}_i, \quad \mathbf{c}_g = \sum_{i=1}^k \mathbf{R}_i^T \mathbf{c}_i \quad (7)$$

- For a weighted **multiplication** gate $g(x_1, \dots, x_k) = \alpha x_1 \cdot \dots \cdot x_k$ do:
For $i \in [k]$ generate matrices $\mathbf{R}_i \in \mathbb{Z}_q^{m \times m}$ such that

$$\mathbf{G}\mathbf{R}_1 = \alpha \mathbf{G} : \mathbf{R}_1 = \text{BD}(\alpha \mathbf{G}) \quad (8)$$

$$\mathbf{G}\mathbf{R}_i = -\mathbf{B}_{i-1} \mathbf{R}_{i-1} : \mathbf{R}_i = \text{BD}(-\mathbf{B}_{i-1} \mathbf{R}_{i-1}) \quad \text{for all } i \in \{2, 3, \dots, k\} \quad (9)$$

Output the following matrices and the ciphertext:

$$\mathbf{B}_g = \mathbf{B}_k \mathbf{R}_k, \quad \mathbf{S}_g = \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i^* \right) \mathbf{S}_j \mathbf{R}_j, \quad \mathbf{c}_g = \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i \right) \mathbf{R}_j^T \mathbf{c}_j \quad (10)$$

For example, for $k = 2$, $\mathbf{B}_g = \mathbf{B}_2 \mathbf{R}_2$, $\mathbf{S}_g = x_2^* \mathbf{S}_1 \mathbf{R}_1 + \mathbf{S}_2 \mathbf{R}_2$, $\mathbf{c}_g = x_2^* \mathbf{R}_1^T \mathbf{c}_1 + \mathbf{R}_2^T \mathbf{c}_2$.

For multiplication gates, the reason we need an upper bound p on all but one of the inputs x_i is that these x_i values are used in (10) and we need the norm of \mathbf{S}_g and the norm of the noise in the ciphertext \mathbf{c}_g to be bounded from above. The next two lemmas show that these algorithms satisfy the required properties to be *FKHE-enabling*.

Lemma 4.4. *Let $\beta_g(m) = km$. For a weighted addition gate $g(\mathbf{x}) = \alpha_1 x_1 + \dots + \alpha_k x_k$ we have:*

1. If $\mathbf{c}_i \in E_{\mathbf{s}, \delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$, then $\mathbf{c}_g \in E_{\mathbf{s}, \Delta}(g(\mathbf{x}), \mathbf{B}_g)$ where $\Delta \leq \beta_g(m) \cdot \delta$ and $\mathbf{B}_g = \text{Eval}_{pk}(g, (\mathbf{B}_1, \dots, \mathbf{B}_k))$.
2. The output \mathbf{S}_g satisfies $\mathbf{A}\mathbf{S}_g - g(\mathbf{x}^*)\mathbf{G} = \mathbf{B}_g$ where $\|\mathbf{S}_g\|_2 \leq \beta_g(m) \cdot \max_{i \in [k]} \|\mathbf{S}_i\|_2$ and $\mathbf{B}_g = \text{Eval}_{pk}(g, (\mathbf{A}\mathbf{S}_1 - x_1^*\mathbf{G}, \dots, \mathbf{A}\mathbf{S}_k - x_k^*\mathbf{G}))$.

Proof. By Eq. 7 the output ciphertext is computed as follows:

$$\begin{aligned}
\mathbf{c}_g &= \sum_{i=1}^k \mathbf{R}_i^T \cdot \mathbf{c}_i = \sum_{i=1}^k \mathbf{R}_i^T \cdot \left((x_i \mathbf{G} + \mathbf{B}_i)^T \mathbf{s} + \mathbf{e}_i \right) = \quad // \text{ substitute for } \mathbf{c}_i = (x_i \mathbf{G} + \mathbf{B}_i)^T \mathbf{s} + \mathbf{e}_i \\
&= \sum_{i=1}^k (x_i \mathbf{G} \mathbf{R}_i)^T \mathbf{s} + \sum_{i=1}^k (\mathbf{B}_i \mathbf{R}_i)^T \mathbf{s} + \sum_{i=1}^k (\mathbf{R}_i^T \mathbf{e}_i) = \quad // \text{ break the product into components} \\
&= \left(\sum_{i=1}^k \alpha_i x_i \right) \mathbf{G}^T \mathbf{s} + \mathbf{B}_g^T \mathbf{s} + \mathbf{e}_g = \quad // \mathbf{G} \mathbf{R}_i = \alpha_i \mathbf{R}_i \text{ from Eq. 6 and } \mathbf{B}_g = \sum_{i=1}^k \mathbf{B}_i \mathbf{R}_i \text{ from Eq. 7} \\
&= [g(\mathbf{x})\mathbf{G} + \mathbf{B}_g]^T \mathbf{s} + \mathbf{e}_g
\end{aligned}$$

The noise bound is: $\|\mathbf{e}_g\| = \|\mathbf{R}_1^T \mathbf{e}_1 + \dots + \mathbf{R}_k^T \mathbf{e}_k\| \leq k \cdot \max_{j \in [k]} \left(\|\mathbf{R}_j^T\|_2 \cdot \|\mathbf{e}_j\| \right) \stackrel{\text{Lemma 2.4, part 4}}{\leq} km \cdot \delta$. This completes the proof of the first part of the lemma.

In the second part of the lemma, by Eq. 7 the output matrix \mathbf{B}_g is computed as follows:

$$\begin{aligned}
\mathbf{B}_g &= \sum_{i=1}^k (\mathbf{A}\mathbf{S}_i - x_i^* \mathbf{G}) \mathbf{R}_i = \quad // \text{ plug-in matrices given in the lemma into Eq. 7} \\
&= \mathbf{A} \sum_{i=1}^k \mathbf{S}_i \mathbf{R}_i - \sum_{i=1}^k \alpha_i x_i^* \mathbf{G} = \mathbf{A}\mathbf{S}_g - g(x^*)\mathbf{G} \quad // \mathbf{G} \mathbf{R}_i = \alpha_i \mathbf{R}_i \text{ from Eq. 6}
\end{aligned}$$

Then $\|\mathbf{S}_g\|_2 = \|\sum_{i=1}^k \mathbf{S}_i \mathbf{R}_i\|_2 \leq k \cdot \max_{i \in [k]} (\|\mathbf{S}_i\|_2 \cdot \|\mathbf{R}_i\|_2) \stackrel{\text{Lemma 2.4, part 4}}{\leq} km \cdot \max_{i \in [k]} (\|\mathbf{S}_i\|_2)$ as required. \square

The next Lemma proves similar bounds for a multiplication gate.

Lemma 4.5. For a **multiplication** gate $g(\mathbf{x}) = \alpha \prod_{i=1}^k x_i$ we have the same bounds on \mathbf{c}_g and \mathbf{S}_g as in Lemma 4.4 with $\beta_g(m) = \frac{p^k - 1}{p - 1} m$.

Proof. Set $\mathbf{e}_g = \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i \right) \mathbf{R}_j^T \mathbf{e}_j$. Then the output ciphertext is computed as follows:

$$\begin{aligned}
\mathbf{c}_g &= \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i \right) \mathbf{R}_j^T \mathbf{c}_j = \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i \right) \mathbf{R}_j^T \left((x_j \mathbf{G} + \mathbf{B}_j)^T \mathbf{s} + \mathbf{e}_j \right) = \quad // \text{ substitute for } \mathbf{c}_j \\
&= \left[\left(\prod_{i=1}^k x_i \right) \mathbf{G} \mathbf{R}_1 + \sum_{j=2}^k \left(\prod_{i=j}^k x_i \right) \cancel{(\mathbf{G} \mathbf{R}_j + \mathbf{B}_{j-1} \mathbf{R}_{j-1})} + \mathbf{B}_k \mathbf{R}_k \right]^T \mathbf{s} + \mathbf{e}_g = \quad // \text{ regroup} \\
&= \left[\left(\prod_{i=1}^k x_i \right) \mathbf{G} \mathbf{R}_1 + \mathbf{B}_k \mathbf{R}_k \right]^T \mathbf{s} + \mathbf{e}_g = \quad // \text{ use Eq. 9 to cancel terms} \\
&= [g(\mathbf{x}) \mathbf{G} + \mathbf{B}_g]^T \mathbf{s} + \mathbf{e}_g \quad // \text{ use the facts } \mathbf{G} \mathbf{R}_1 = \alpha \mathbf{G} \text{ (Eq. 8), } \mathbf{B}_g = \mathbf{B}_k \mathbf{R}_k \text{ (Eq. 10)}
\end{aligned}$$

The bound on the noise $\|\mathbf{e}_g\|$ is:

$$\|\mathbf{e}_g\| = \left\| \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i \right) \mathbf{R}_j^T \mathbf{e}_j \right\| \leq \left(1 + p + \dots + p^{k-1} \right) \cdot \max_{j \in [k]} [\|\mathbf{R}_j^T\|_2 \cdot \|\mathbf{e}_j\|] \stackrel{\text{Lemma. 2.3}}{\leq} \frac{p^k - 1}{p - 1} m \cdot \delta$$

This completes the first part of the lemma. In the second part of the lemma, the output matrix \mathbf{B}_g is computed as follows:

$$\begin{aligned}
\mathbf{B}_g &= (\mathbf{A} \mathbf{S}_k - x_k \mathbf{G}) \mathbf{R}_k \stackrel{\text{Eq. 9}}{=} \quad // \text{ by (9) we have } \mathbf{G} \mathbf{R}_k = -(\mathbf{A} \mathbf{S}_{k-1} - x_{k-1} \mathbf{G}) \mathbf{R}_{k-1} \\
&= (\mathbf{A} \mathbf{S}_k \mathbf{R}_k + x_k \mathbf{A} \mathbf{S}_{k-1} \mathbf{R}_{k-1} - x_k \cdot x_{k-1} \mathbf{G} \mathbf{R}_{k-1}) \stackrel{\text{Eq. 9}}{=} \dots \stackrel{\text{Eq. 9}}{=} \\
&= (\mathbf{A} \mathbf{S}_k \mathbf{R}_k + x_k \mathbf{A} \mathbf{S}_{k-1} \mathbf{R}_{k-1} + x_k \cdot x_{k-1} \mathbf{A} \mathbf{S}_{k-2} \mathbf{R}_{k-2} + \dots + (-x_1 \dots x_k \mathbf{G} \mathbf{R}_1)) \stackrel{\text{Eq. 8}}{=} \\
&= (\mathbf{A} \mathbf{S}_g - \alpha x_1 \dots x_k \mathbf{G}) = (\mathbf{A} \mathbf{S}_g - g(\mathbf{x}) \mathbf{G})
\end{aligned}$$

Moreover, the bound on the norm of \mathbf{S}_g is:

$$\begin{aligned}
\|\mathbf{S}_g\|_2 &= \left\| \sum_{j=1}^k \left(\prod_{i=j+1}^k x_i \right) \mathbf{S}_j \mathbf{R}_j \right\|_2 \\
&\leq \left(1 + p + \dots + p^{k-1} \right) \max_{i \in [k]} (\|\mathbf{S}_i\|_2 \cdot \|\mathbf{R}_i\|_2) \stackrel{\text{Lemma. 2.3}}{\leq} \frac{p^k - 1}{p - 1} m \cdot \max_{i \in [k]} (\|\mathbf{S}_i\|_2)
\end{aligned}$$

as required. \square

4.3 Evaluation algorithms for circuits

We will now show how using the algorithms for single gates, that compute weighted additions and multiplications as described above, to build algorithms for the depth d , unbounded fan-in circuits.

Let $\{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of polynomial-size arithmetic circuits. For each $\mathcal{C} \in \mathcal{C}_\lambda$ we index the wires of \mathcal{C} following the notation in [GVW13]. The input wires are indexed 1 to ℓ , the internal wires have indices $\ell + 1, \ell + 2, \dots, |\mathcal{C}| - 1$ and the output wire has index $|\mathcal{C}|$, which also denotes the size of the circuit. Every gate $g_w : \mathbb{Z}_q^{k_w} \rightarrow \mathbb{Z}_q$ (in \mathcal{G} as per 5) is indexed as a tuple (w_1, \dots, w_{k_w}, w)

where k_w is the fan-in of the gate. We assume that all (but possibly one) of the input values to the multiplication gates are bounded by p which is smaller than scheme modulus q . The “fan-out wires” in the circuit are given a single number. That is, if the outgoing wire of a gate feeds into the input of multiple gates, then all these wires are indexed the same. For some $\lambda \in \mathbb{N}$, define the family of functions $\mathcal{F} = \{f : f \text{ can be computed by some } \mathcal{C} \in \mathcal{C}_\lambda\}$. Again we will describe the three Eval algorithms together, but it is easy to see that they can be separated.

$$\begin{aligned} \text{Eval}_{\text{pk}}(f \in \mathcal{F}, \vec{\mathbf{B}} \in (\mathbb{Z}_q^{n \times m})^\ell) &\longrightarrow \mathbf{B}_f \in \mathbb{Z}_q^{n \times m} \\ \text{Eval}_{\text{ct}}(f \in \mathcal{F}, ((x_i, \mathbf{B}_i, \mathbf{c}_i))_{i=1}^\ell) &\longrightarrow \mathbf{c}_f \in \mathbb{Z}_q^m \\ \text{Eval}_{\text{sim}}(f \in \mathcal{F}, ((x_i^*, \mathbf{S}_i))_{i=1}^\ell, \mathbf{A}) &\longrightarrow \mathbf{S}_f \in \mathbb{Z}_q^{m \times m} \end{aligned}$$

Let f be computed by some circuit $\mathcal{C} \in \mathcal{C}_\lambda$, that has ℓ input wires. We construct the required matrices inductively input to output gate-by-gate.

For all $w \in [\mathcal{C}]$ denote the value that wire w carries when circuit \mathcal{C} is evaluated on \mathbf{x} or \mathbf{x}^* to be x_w or x_w^* respectively. Consider an arbitrary gate of fan-in k_w (we will omit the subscript w where it is clear from the context): (w_1, \dots, w_k, w) that computes the function $g_w : \mathbb{Z}_q^k \rightarrow \mathbb{Z}_q$. Each wire w_i carries a value x_{w_i} . Suppose we already computed $\mathbf{B}_{w_1}, \dots, \mathbf{B}_{w_k}, \mathbf{S}_{w_1}, \dots, \mathbf{S}_{w_k}$ and $\mathbf{c}_{w_1}, \dots, \mathbf{c}_{w_k}$, note that if w_1, \dots, w_k are all in $\{1, 2, \dots, \ell\}$ then these matrices and vectors are the inputs of the corresponding Eval functions.

Using Eval algorithms described in Section 4.2, compute

$$\begin{aligned} \mathbf{B}_w &= \text{Eval}_{\text{pk}}(g_w, (\mathbf{B}_{w_1}, \dots, \mathbf{B}_{w_k})) \\ \mathbf{c}_w &= \text{Eval}_{\text{ct}}(g_w, ((x_{w_i}, \mathbf{B}_{w_i}, \mathbf{c}_{w_i}))_{i=1}^k) \\ \mathbf{S}_w &= \text{Eval}_{\text{sim}}(g_w, ((x_{w_i}^*, \mathbf{S}_{w_i}))_{i=1}^k, \mathbf{A}) \end{aligned}$$

Output $\mathbf{B}_f := \mathbf{B}_{|\mathcal{C}|}$, $\mathbf{c}_f := \mathbf{c}_{|\mathcal{C}|}$, $\mathbf{S}_f := \mathbf{S}_{|\mathcal{C}|}$. Next we show that these outputs satisfy the required properties.

Lemma 4.6. *Let $\beta(m) = \frac{p-1}{p-1}m$. If $\mathbf{c}_i \in E_{\mathbf{s}, \delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $\delta > 0$, then $\mathbf{c}_f \in E_{\mathbf{s}, \Delta}(f(\mathbf{x}), \mathbf{B}_f)$ where $\Delta < (\beta(m))^d \cdot \delta$ and $\mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$.*

Proof. By Lemma 4.4 and 4.5, after each level of the circuit the noise is multiplied by $\beta_{g_w}(m)$, which is upperbounded by $\beta(m)$ and the total number of levels is equal to the depth d of the circuit. The lemma follows. \square

Lemma 4.7. *Let $\beta(m)$ be as defined in Lemma 4.6. If $\mathbf{S}_1, \dots, \mathbf{S}_\ell$ are random matrices in $\{\pm 1\}^{m \times m}$, then the output \mathbf{S}_f satisfies $\mathbf{A}\mathbf{S}_f - f(\mathbf{x}^*)\mathbf{G} = \mathbf{B}_f$ where $\|\mathbf{S}_f\|_2 \leq (\beta(m))^d \cdot 20\sqrt{m}$ and $\mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{A}\mathbf{S}_1 - x_1^*\mathbf{G}, \dots, \mathbf{A}\mathbf{S}_\ell - x_\ell^*\mathbf{G}))$.*

Proof. Since the input \mathbf{S}_i for $i \in [\ell]$ are random matrices in $\{\pm 1\}^{m \times m}$, by Lemma 2.5 for all $i \in [\ell]$, $\|\mathbf{S}_i\|_2 < 20\sqrt{m}$. By Lemma 4.4 and 4.5, after each level of the circuit the bound on \mathbf{S} gets multiplied by at most $\beta(m)$, therefore after d levels, which is the depth of the circuit, the bound on the output matrix will be $\|\mathbf{S}_f\|_2 \leq (\beta(m))^d \cdot 20\sqrt{m}$. The lemma follows. \square

In summary, algorithms $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ are $\alpha_{\mathcal{F}}$ -FKHE enabling for

$$\alpha_{\mathcal{F}}(n) = (\beta(m))^d \cdot 20\sqrt{m} = O((p^{k-1}m)^d \sqrt{m}), \quad \text{where } m = \Theta(n \log q). \quad (11)$$

This is sufficient for polynomial depth arithmetic circuits as discussed in the introduction.

4.4 ABE with Short Secret Keys for Arithmetic Circuits from LWE

The FKHE for a family of functions $\mathcal{F} = \{f : (\mathbb{Z}_q)^\ell \rightarrow \mathbb{Z}_q\}$ we constructed in Section 4 immediately gives a key-policy ABE as discussed in Section 3. For completeness we briefly describe the resulting ABE system.

Given *FKHE-enabling* algorithms $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ for a family of functions \mathcal{F} from Section 4.1, the ABE system works as follows:

- **Setup** $(1^\lambda, \ell)$: Choose n, q, χ, m and σ as in “Parameters” in Section 4.
Run algorithm $\text{TrapGen}(1^n, 1^m, q)$ (Lemma 2.4, part 1) to generate $(\mathbf{A}, \mathbf{T}_\mathbf{A})$.
Choose random matrices $\mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell \in \mathbb{Z}_q^{n \times m}$ and output the keys:

$$\text{mpk} = (\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell) \quad ; \quad \text{msk} = (\mathbf{T}_\mathbf{A}, \mathbf{D}, \mathbf{B}_1, \dots, \mathbf{B}_\ell)$$

- **Keygen** (msk, f) : Let $\mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$.
Output $\text{sk}_f := \mathbf{R}_f$ where \mathbf{R}_f is a low-norm matrix in $\mathbb{Z}^{2m \times m}$ sampled from the discrete Gaussian distribution $\mathcal{D}_\sigma(\Lambda_q^\mathbf{D}(\mathbf{A}|\mathbf{B}_f))$ so that $(\mathbf{A}|\mathbf{B}_f) \cdot \mathbf{R}_f = \mathbf{D}$.
To construct \mathbf{R}_f run algorithm $\text{SampleRight}(\mathbf{A}, \mathbf{T}_\mathbf{A}, y\mathbf{G} + \mathbf{B}_f, \mathbf{D}, \sigma)$ from Lemma 2.8, part 1.
Note that the secret key sk_f is always in $\mathbb{Z}^{2m \times m}$ independent of the complexity of the function f .
- **Enc** $(\text{mpk}, \mathbf{x} \in \mathbb{Z}_q^\ell, \mu \in \{0, 1\}^m)$: Choose a random vector $\mathbf{s} \leftarrow \mathbb{Z}_q^n$ and error vectors $\mathbf{e}_0, \mathbf{e}_1 \leftarrow \chi^m$. Choose ℓ uniformly random matrices $\mathbf{S}_i \leftarrow \{\pm 1\}^{m \times m}$ for $i \in [\ell]$. Set

$$\begin{aligned} \mathbf{H} &= (\mathbf{A} \mid x_1\mathbf{G} + \mathbf{B}_1 \mid \dots \mid x_\ell\mathbf{G} + \mathbf{B}_\ell) \in \mathbb{Z}_q^{n \times (\ell+1)m} \\ \mathbf{e} &= (\mathbf{I}_m | \mathbf{S}_1 | \dots | \mathbf{S}_\ell)^\top \cdot \mathbf{e}_0 \in \mathbb{Z}_q^{(\ell+1)m} \end{aligned}$$

Output $\mathbf{c} = (\mathbf{H}^\top \mathbf{s} + \mathbf{e}, \mathbf{D}^\top \mathbf{s} + \mathbf{e}_1 + \lceil q/2 \rceil \mu) \in \mathbb{Z}_q^{(\ell+2)m}$.

- **Dec** $(\text{sk}_f, (\mathbf{x}, \mathbf{c}))$: If $f(\mathbf{x}) \neq 0$ output \perp . Otherwise, let the ciphertext $\mathbf{c} = (\mathbf{c}_{\text{in}}, \mathbf{c}_1, \dots, \mathbf{c}_\ell, \mathbf{c}_{\text{out}}) \in \mathbb{Z}_q^{(\ell+2)m}$, set $\mathbf{c}_f = \text{Eval}_{\text{ct}}(f, \{(x_i, \mathbf{B}_i, \mathbf{c}_i)\}_{i=1}^\ell) \in \mathbb{Z}_q^m$.
Let $\mathbf{c}'_f = (\mathbf{c}_{\text{in}} | \mathbf{c}_f) \in \mathbb{Z}_q^{2m}$ and output $\text{Round}(\mathbf{c}_{\text{out}} - \mathbf{R}_f^\top \mathbf{c}'_f) \in \{0, 1\}^m$.

This completes the description of the system. The proof of the following security theorem follows from Theorems 4.2 and 3.2.

Theorem 4.8. *For FKHE-enabling algorithms $(\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}})$ for the family of functions \mathcal{F} , the ABE system above is correct and selectively-secure with respect to \mathcal{F} , assuming the (n, q, χ) -LWE assumption holds where n, q, χ are the parameters for the FKHE-enabling algorithms.*

5 Extensions

5.1 Key Delegation

Our ABE easily extends to support full key delegation. We first sketch the main idea for adding key delegation and then describe the resulting ABE system.

In the ABE scheme from Section 4.4, a secret key for a function f is a matrix \mathbf{R}_f that maps $(\mathbf{A}|\mathbf{B}_f)$ to some fixed matrix \mathbf{D} . Instead, we can give as a secret key for f a trapdoor (i.e. a short basis) \mathbf{T}_F for the matrix $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f)$. The decryptor could use \mathbf{T}_F to generate the matrix \mathbf{R}_f herself using algorithm `SampleD`. Now, for a given function g , to construct a secret key that decrypts whenever the attribute vector \mathbf{x} satisfies $f(\mathbf{x}) = g(\mathbf{x}) = 0$ we extend the trapdoor for \mathbf{F} into a trapdoor for $(\mathbf{F}|\mathbf{B}_g) = (\mathbf{A}|\mathbf{B}_f|\mathbf{B}_g)$ using algorithm `ExtendRight`. We give a randomized version of this trapdoor as a delegated secret key for $f \wedge g$. Intuitively this trapdoor can only be used to decrypt if the decryptor can obtain the ciphertexts under matrices \mathbf{B}_f and \mathbf{B}_g which by security of ABE can only happen if the ciphertexts was created for an attribute vector \mathbf{x} satisfying $f(\mathbf{x}) = g(\mathbf{x}) = 0$.

The top level secret key generated by `Keygen` is a $(2m \times 2m)$ matrix in \mathbb{Z} . After k delegations the secret key becomes a $((k+1)m \times (k+1)m)$ matrix. Hence, the delegated key grows quadratically with the number of delegations k .

Definition. Formally, a delegatable attribute-based encryption (DABE) scheme is an attribute-based encryption scheme that in addition to four standard algorithms (`Setup`, `Keygen`, `Enc`, `Dec`) offers a delegation algorithm `Delegate`. Consider a ciphertext c encrypted for index vector \mathbf{x} . The algorithm `Keygen` returns the secret key sk_f for function f and this key allows to decrypt the ciphertext c only if $f(\mathbf{x}) = 0$. The delegation algorithm given the key sk_f and a function g outputs a “delegated” secret key that allows to decrypt the ciphertext only if $f(\mathbf{x}) = 0 \wedge g(\mathbf{x}) = 0$, which is a more restrictive condition. The idea can be generalized to arbitrary number of delegations:

`Delegate`($\text{mpk}, sk_{f_1, \dots, f_k}, f_{k+1}$) $\rightarrow sk_{f_1, \dots, f_{k+1}}$:

Takes as input the master secret key msk , the function $f_{k+1} \in \mathcal{F}$ and the secret key sk_{f_1, \dots, f_k} that was generated either by algorithm `Keygen`, if $k = 1$ or by algorithm `Delegate`, if $k > 1$.

Outputs a secret key $sk_{f_1, \dots, f_{k+1}}$.

Correctness. We require the scheme to give a correct ABE as discussed in Section 2.1 and in addition to satisfy the following requirement. For all sequence of functions $f_1, \dots, f_k \in \mathcal{F}$, a message $m \in \mathcal{M}$ and index $\mathbf{x} \in \mathbb{Z}_q^\ell$, s.t. $f_1(\mathbf{x}) = 0 \wedge \dots \wedge f_k(\mathbf{x}) = 0$ it holds that $\mu = \text{Dec}(sk_{f_1, \dots, f_k}, (\mathbf{x}, c))$ with an overwhelming probability over the choice of $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(1^\lambda, \ell)$, $c \leftarrow \text{Enc}(\text{mpk}, \mathbf{x} \in \mathcal{X}^\ell, \mu)$, $sk_{f_1} \leftarrow \text{Keygen}(\text{msk}, f_1)$ and $sk_{f_1, \dots, f_{i+1}} \leftarrow \text{Delegate}(\text{mpk}, sk_{f_1, \dots, f_i}, f_{i+1})$ for all $i \in [k]$.

Security. The security of DABE schemes is derived from definition of selective security for ABE scheme (see Definition 2.1) by providing the adversary with access to a key-generation oracle.

Definition 5.1 (Selectively-secure DABE). A DABE scheme $\Pi = (\text{Setup}, \text{Keygen}, \text{Enc}, \text{Dec}, \text{Delegate})$ for a class of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ with $\ell = \ell(\lambda)$ inputs over an index space $\mathcal{X}^\ell = \{\mathcal{X}_\lambda^\ell\}_{\lambda \in \mathbb{N}}$ and a message space $\mathcal{M} = \{\mathcal{M}_\lambda\}_{\lambda \in \mathbb{N}}$ is *selectively secure* if for any probabilistic polynomial-time adversary \mathcal{A} , there exists a negligible function $\nu(\lambda)$ such that

$$\text{Adv}_{\Pi, \mathcal{A}}^{\text{sDABE}}(\lambda) \stackrel{\text{def}}{=} \left| \Pr \left[\text{Expt}_{\text{sDABE}, \Pi, \mathcal{A}}^{(0)}(\lambda) = 1 \right] - \Pr \left[\text{Expt}_{\text{sDABE}, \Pi, \mathcal{A}}^{(1)}(\lambda) = 1 \right] \right| \leq \nu(\lambda),$$

where for each $b \in \{0, 1\}$ and $\lambda \in \mathbb{N}$ the experiment $\text{Expt}_{\text{sDABE}, \Pi, \mathcal{A}}^{(b)}(\lambda)$ is defined as follows:

1. $(\mathbf{x}^*, \text{state}_1) \leftarrow \mathcal{A}(\lambda)$, where $\mathbf{x}^* \in \mathcal{X}^\ell$.
2. $(\text{mpk}, \text{msk}) \leftarrow \text{Setup}(\lambda)$.
3. $(\mu_0, \mu_1, \text{state}_2) \leftarrow \mathcal{A}^{\text{KG}(\text{msk}, \mathbf{x}^*, \cdot)}(\text{mpk}, \text{state}_1)$, where $\mu_0, \mu_1 \in \mathcal{M}_\lambda$.
4. $c^* \leftarrow \text{Enc}(\text{mpk}, \mathbf{x}^*, \mu_b)$.
5. $b' \leftarrow \mathcal{A}^{\text{KG}(\text{msk}, \mathbf{x}^*, \cdot)}(c^*, \text{state}_2)$.
6. Output $b' \in \{0, 1\}$.

Here the key-generation oracle $\text{KG}(\text{msk}, \mathbf{x}^*, (f_1, \dots, f_k))$ takes a set of functions $f_1, \dots, f_k \in \mathcal{F}$ and returns the secret key $\text{sk}_{f_1, \dots, f_k}$ if $f_1(\mathbf{x}^*) \neq 0 \vee \dots \vee f_k(\mathbf{x}^*) \neq 0$ and otherwise the oracle returns \perp . The secret key $\text{sk}_{f_1, \dots, f_k}$ is defined as follows: $\text{sk}_{f_1} = \text{KeyGen}(\text{msk}, f_1)$ and for all $i \in \{2, \dots, k\}$ $\text{sk}_{f_1, \dots, f_i} = \text{Delegate}(\text{mpk}, \text{sk}_{f_1, \dots, f_{i-1}}, f_i)$.

5.1.1 A delegatable ABE scheme from LWE

The DABE scheme will be almost identical to ABE scheme described earlier, except as a secret key for function f instead of recoding matrix from $(\mathbf{A}|\mathbf{B}_f)$ to \mathbf{D} we will give the rerandomized trapdoor for $(\mathbf{A}|\mathbf{B}_f)$ and then the decryptor can build the recoding matrix to \mathbf{D} himself.

KeyGen(msk, f) :

Let $\mathbf{B}_f = \text{Eval}_{\text{pk}}(f, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$.

Build the basis \mathbf{T}_f for $\mathbf{F} = (\mathbf{A}|\mathbf{B}_f) \in \mathbb{Z}_q^{n \times 2m}$ as $\mathbf{T}_f \leftarrow \text{RandBasis}(\mathbf{F}, \text{ExtendRight}(\mathbf{A}, \mathbf{T}_A, \mathbf{B}_f), \sigma)$, for big enough $\sigma = \|\mathbf{T}_A\|_{\text{gs}} \cdot \omega(\sqrt{\log m})$ (we will set σ as before: $\sigma = \omega(\alpha_{\mathcal{F}} \cdot \sqrt{\log m})$).

Output $\text{sk}_f := \mathbf{T}_f$.

Delegate(mpk, $\text{sk}_{f_1, \dots, f_k}, g$) :

Parse the secret key $\text{sk}_{f_1, \dots, f_k}$ as a matrix $\mathbf{T}_k \in \mathbb{Z}_q^{(k+1)m \times (k+1)m}$ which is a trapdoor for the matrix $(\mathbf{A}|\mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k}) \in \mathbb{Z}_q^{n \times (k+1)m}$.

Let $\mathbf{B}_g = \text{Eval}_{\text{pk}}(g, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$.

Build the basis for matrix $\mathbf{F} = (\mathbf{A}|\mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k} | \mathbf{B}_g) \in \mathbb{Z}_q^{n \times (k+2)m}$:

$$\mathbf{T}_{k+1} = \text{RandBasis}(\mathbf{F}, \text{ExtendRight}((\mathbf{A}|\mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k}), \mathbf{T}_k, \mathbf{B}_g), \sigma_k).$$

Here $\sigma_k = \sigma \cdot (\sqrt{m \log m})^k$. Output $\text{sk}_{f_1, \dots, f_k, g} := \mathbf{T}_{k+1} \in \mathbb{Z}_q^{(k+2)m \times (k+2)m}$. Note that the size of the key grows quadratically with the number of delegations k .

Dec($\text{sk}_{f_1, \dots, f_k}, (\mathbf{x}, \mathbf{c})$) : If $f_1(\mathbf{x}) \neq 0 \vee \dots \vee f_k(\mathbf{x}) \neq 0$ output \perp .

Otherwise parse the secret key $\text{sk}_{f_1, \dots, f_k}$ as a matrix $\mathbf{T}_k \in \mathbb{Z}_q^{(k+1)m \times (k+1)m}$ which is a trapdoor for the matrix $(\mathbf{A}|\mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k})$.

Run $\mathbf{R} \leftarrow \text{SampleD}((\mathbf{A}|\mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k}), \mathbf{T}_k, \mathbf{D}, \sigma_k)$ to generate a low-norm matrix

$\mathbf{R} \in \mathbb{Z}_q^{(k+1)m \times m}$ such that $(\mathbf{A}|\mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k}) \cdot \mathbf{R} = \mathbf{D}$.

For all $j \in [k]$, compute $(\mathbf{c}_{in}, \mathbf{c}_j, \mathbf{c}_{out}) = \text{Eval}_{\text{ct}}(\{(x_i, \mathbf{B}_i)\}_{i=1}^\ell, \mathbf{c}, f_j) \in \mathbb{Z}_q^{3m}$. Note that \mathbf{c}_{in} and \mathbf{c}_{out} stay the same across all $i \in [k]$.

Let $\mathbf{c}' = (\mathbf{c}_{in} | \mathbf{c}_1 | \dots | \mathbf{c}_k) \in \mathbb{Z}_q^{(k+1)m}$. Output $\mu = \text{Round}(\mathbf{c}_{out} - \mathbf{R}^T \mathbf{c}')$.

Correctness. To show correctness, note that when $f_1(\mathbf{x}) = 0 \wedge \dots \wedge f_k(\mathbf{x}) = 0$ we know by the requirement on Eval_{ct} that the resulting ciphertexts $\mathbf{c}_{f_i} \in \mathbf{E}_{\mathbf{s}, \Delta}(0, \mathbf{B}_{f_i})$ for $\forall i \in [k]$. Consequently,

$$(\mathbf{c}_{in} | \mathbf{c}_{f_1} | \dots | \mathbf{c}_{f_k}) = (\mathbf{A} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k})^T \mathbf{s} + \mathbf{e}' \quad \text{where} \quad \|\mathbf{e}'\| < k\Delta + \chi_{\max} < (k\alpha_{\mathcal{F}} + 1)\chi_{\max}.$$

We know that $(\mathbf{A} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k}) \cdot \mathbf{R} = \mathbf{D}$ and $\|\mathbf{R}^T\|_2 < (k+1)m\sigma_k$ with overwhelming probability by Lemma 2.5. Therefore

$$\mathbf{c}_{out} - \mathbf{R}^T \mathbf{c}'_f = (\mathbf{D}^T \mathbf{s} + \mathbf{e}_1) - (\mathbf{D}^T \mathbf{s} + \mathbf{R}^T \mathbf{e}') = \mathbf{e}_1 - \mathbf{R}^T \mathbf{e}'.$$

Finally,

$$\|\mathbf{e}_1 - \mathbf{R}^T \mathbf{e}'\| \leq \chi_{\max} + (k+1)m\sigma_k \cdot (\alpha_{\mathcal{F}} + 1)\chi_{\max} \leq (k+2)\alpha_{\mathcal{F}}^2 \cdot \chi_{\max} \cdot m^{k/2+1}$$

with overwhelming probability. The bound on $\alpha_{\mathcal{F}} : \alpha_{\mathcal{F}}^2 m^{k/2+1} < \frac{1}{4(k+2)} \cdot (q/\chi_{\max})$ ensures that this quantity is less than $q/4$ thereby ensuring correct decryption of all bits of $\mu \in \{0, 1\}^m$.

Security. The security game is similar to the security game for FKHE, described in Section 4, except in Game 2 we need to answer delegated key queries. Consider a private key query $\text{sk}_{f_1, \dots, f_k}$, where $f_1, \dots, f_k \in \mathcal{F}$. This query is only allowed when $f_1(\mathbf{x}^*) \neq 0 \vee \dots \vee f_k(\mathbf{x}^*) \neq 0$. Without loss of generality, assume that $f_1(\mathbf{x}^*) = 0 \wedge \dots \wedge f_{k-1}(\mathbf{x}^*) = 0$ and $f_k(\mathbf{x}^*) \neq 0$. Indeed for all other cases, the adversary may ask for the key for a smaller sequence of functions and delegate herself. The key generation oracle for all $i \in [k]$ computes $\mathbf{B}_{f_i} = \text{Eval}_{\text{pk}}(f_i, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$ and needs to produce a trapdoor $\mathbf{T}_k \in \mathbb{Z}^{(k+1)m \times (k+1)m}$ for the matrix $(\mathbf{A} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k}) \in \mathbb{Z}_q^{n \times (k+1)m}$.

To do so the key generation oracle does:

- Run $\mathbf{S}_{f_k} \leftarrow \text{Eval}_{\text{sim}}(f_k, ((x_i^*, \mathbf{S}_i^*))_{i=1}^\ell, \mathbf{A})$ and obtains a low-norm matrix $\mathbf{S}_{f_k} \in \mathbb{Z}_q^{m \times m}$ such that $\mathbf{A} \mathbf{S}_{f_k} - f_k(\mathbf{x}^*) \mathbf{G} = \mathbf{B}_{f_k}$. By definition of Eval_{sim} we know that $\|\mathbf{S}_{f_k}\|_2 \leq \alpha_{\mathcal{F}}$.
- Let $\mathbf{F} = (\mathbf{A} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k}) = (\mathbf{A} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_{k-1}} | \mathbf{A} \mathbf{S}_{f_k} - y^* \mathbf{G})$. Because $y^* \neq 0$ the key generation oracle can obtain a trapdoor $\mathbf{T}_{(\mathbf{A} | \mathbf{B}_{f_k})}$ by running

$$\mathbf{T}_{(\mathbf{A} | \mathbf{B}_{f_k})} \leftarrow \text{ExtendLeft}(y^* \mathbf{G}, \mathbf{T}_{\mathbf{G}}, \mathbf{A}, \mathbf{S}_{f_k})$$

And then produce $\mathbf{T}_{(\mathbf{A} | \mathbf{B}_{f_k} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_{k-1}})}$ by running

$$\mathbf{T}_{(\mathbf{A} | \mathbf{B}_{f_k} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_{k-1}})} \leftarrow \text{ExtendRight}(\mathbf{G}, \mathbf{T}_{\mathbf{G}}, (\mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_{k-1}}))$$

Now we can switch the rows of the matrix $\mathbf{T}_{(\mathbf{A} | \mathbf{B}_{f_k} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_{k-1}})}$ to get the matrix $\mathbf{T}_{\mathbf{F}}$, which is a trapdoor for $(\mathbf{A} | \mathbf{B}_{f_1} | \dots | \mathbf{B}_{f_k})$. This operation, as well as ExtendRight function (according to Lemma 2.4, part 2) does not change the Gram-Schmidt norm of the basis, therefore this trapdoor satisfies

$$\|\mathbf{T}_{\mathbf{F}}\|_{\text{gs}} \leq \|\mathbf{T}_{\mathbf{G}}\|_{\text{gs}} \cdot \|\mathbf{S}_{f_k}\|_2 \leq \sqrt{5}\alpha_{\mathcal{F}}(n)$$

where the bound on $\|\mathbf{T}_{\mathbf{G}}\|_{\text{gs}}$ is from Lemma 2.4 (part 4).

- Finally, it responds with rerandomized trapdoor $\mathbf{T}_k = \text{RandBasis}(\mathbf{F}, \mathbf{T}_{\mathbf{F}}, \sigma_k)$. By definition of RandBasis we know that \mathbf{T}_k is distributed as $\mathcal{D}_{\sigma_k}(\Lambda_q^{\mathbf{F}}(\mathbf{F}))$ as required. Indeed $\sigma_k = \|\mathbf{T}_{\mathbf{F}}\|_{\text{gs}} \cdot \omega(\sqrt{\log m})$ as needed for algorithm RandBasis in Lemma 2.6 (part 3).

5.2 Polynomial gates

We can further reduce the depth of a given arithmetic circuit (and thereby shrink the required lattice sizes) by allowing the circuit to use more general gates than simple addition and multiplication. For example, the k -way OR gate polynomial can be implemented using a single gate.

Definition 5.2. An ℓ -variate polynomial is said to have *restricted arithmetic complexity* (ℓ, d, g) if it can be computed by a depth- d circuit that takes ℓ inputs $x_1, \dots, x_\ell \in \mathbb{Z}_q$ and outputs a single $x \in \mathbb{Z}_q$. The circuit contains g gates, each of them is either a fan-in 2 addition gate or a fan-in 2 multiplication gate. Multiplication gates are further restricted to have one of their two inputs be one of the inputs to the circuit: x_1, \dots, x_ℓ .

We build the Eval algorithms for polynomials with complexity (ℓ, d, g) whose running time is proportional to g and that increase the magnitude of the noise in a given ciphertext by a factor of at most $O(p^d \cdot m)$, where p is the bound on all the intermediate values. Were we to directly use the Eval algorithms from the previous section on this polynomial, the magnitude of the noise would increase by $O((pm)^d)$ which is considerably larger, especially when p is small (e.g. $p = 1$).

We can build arithmetic circuits using polynomials with complexity (ℓ, d, g) as gates. Evaluating a depth D arithmetic circuit with such polynomial gates would increase the magnitude of the noise by at most a factor of $O((p^d \cdot m)^D)$. Again, if we were to simply treat the circuit as a standard arithmetic circuit with basic addition and multiplication gates the noise would instead grow as $O((pm)^{dD})$ which is larger.

Next we present ABE-enabling algorithms $\text{Eval}_{\text{pk}}, \text{Eval}_{\text{ct}}, \text{Eval}_{\text{sim}}$ for these enhanced polynomial gates with the noise bounds discussed in the previous paragraph. To support multiplication and addition of constants, we may assume that we have an extra 0-th input to the circuit that always carries the value 1. We present all three algorithms at the same time. Suppose that f is a polynomial with complexity (ℓ, d, g) , then the three algorithms work as follows:

$$\begin{aligned} \text{Eval}_{\text{pk}}(f, \vec{\mathbf{B}} \in (\mathbb{Z}_q^{n \times m})^\ell) &\longrightarrow \mathbf{B}_f \in \mathbb{Z}_q^{n \times m} \\ \text{Eval}_{\text{ct}}(f, ((x_i, \mathbf{B}_i, \mathbf{c}_i))_{i=1}^\ell) &\longrightarrow \mathbf{c}_f \in \mathbb{Z}_q^m \\ \text{Eval}_{\text{sim}}(f, ((x_i, \mathbf{S}_i))_{i=1}^\ell, \mathbf{A}) &\longrightarrow \mathbf{S}_f \in \mathbb{Z}_q^{m \times m} \end{aligned}$$

For each wire $w \in [|f|]$ (here $|f|$ denotes the total number of wires in the circuit and the notation of naming the wires is as described in Section 4.3) starting from the input wires and proceeding to the output we will construct the matrices $\mathbf{B}_w \in \mathbb{Z}_q^{n \times m}$, $\mathbf{S}_w \in \mathbb{Z}_q^{m \times m}$, $\mathbf{c}_w \in \mathbb{Z}_q^m$. Finally we output $\mathbf{B}_f = \mathbf{B}_{|f|}$, $\mathbf{S}_f = \mathbf{S}_{|f|}$, $\mathbf{c}_f = \mathbf{c}_{|f|}$. Consider an arbitrary gate and suppose that matrices on the input wires are computed, then to compute the matrices on the output wire do the following:

- Suppose the gate computes addition, has input wires w_1 and w_2 and output wire w . Then set the output matrices on wire w to be:

$$\mathbf{B}_w = \mathbf{B}_{w_1} + \mathbf{B}_{w_2}, \quad \mathbf{S}_w = \mathbf{S}_{w_1} + \mathbf{S}_{w_2}, \quad \mathbf{c}_w = \mathbf{c}_{w_1} + \mathbf{c}_{w_2}.$$

- Suppose the gate computes the multiplication by x_i for some $i \in [\ell]$, the input wires are u and i , the output wire is w . Then generate matrix $\mathbf{R} \in \mathbb{Z}_q^{m \times m}$ to satisfy $\mathbf{GR} = -\mathbf{B}_u$ by running $\mathbf{R} = \text{BD}(-\mathbf{B}_u)$. Output

$$\mathbf{B}_w = \mathbf{B}_i \mathbf{R}, \quad \mathbf{S}_w = \mathbf{S}_i \mathbf{R} + x_w \mathbf{S}_u, \quad \mathbf{c}_w = x_i \mathbf{c}_u + \mathbf{R}^T \mathbf{c}_i.$$

Note that the amount of work required to run the Eval algorithms is proportional to the number of gates g in the circuit.

The following lemma shows that the noise in the output ciphertext grows by at most the factor of $O(p^d m)$, where p is the upper bound on the intermediate values in the circuit.

Lemma 5.3. *If $\mathbf{c}_i \in E_{\mathbf{s}, \delta}(x_i, \mathbf{B}_i)$ for some $\mathbf{s} \in \mathbb{Z}_q^n$, $\delta > 0$ and the bound on the numbers $p \geq 2$, then for the polynomial f of complexity (ℓ, d, g) with $\beta_d = (1 + p + \dots + p^d) \cdot m$ we have:*

- \mathbf{c}_f satisfies $\mathbf{c}_f \in E_{\mathbf{s}, \Delta}(f(\mathbf{x}), \mathbf{B}_f)$ where $\mathbf{B}_f = \text{Eval}_{pk}(f, (\mathbf{B}_1, \dots, \mathbf{B}_\ell))$ and $\Delta < \beta_d(m) \cdot \delta$,
- \mathbf{S}_f satisfies $\mathbf{A}\mathbf{S}_f - f(\mathbf{x})\mathbf{G} = \mathbf{B}_f$ where $\mathbf{B}_f = \text{Eval}_{pk}(f, (\mathbf{A}\mathbf{S}_1 - x_1\mathbf{G}, \dots, \mathbf{A}\mathbf{S}_\ell - x_\ell\mathbf{G}))$ and $\|\mathbf{S}_f\|_2 \leq \beta_d(m) \cdot \gamma$ where $\gamma = \max_{i \in [\ell]} \|\mathbf{S}_i\|_2$.

Proof. We prove the lemma by induction.

- Consider an addition gate at level i with input wires w_1 and w_2 and output wire w . Suppose for $j \in [2]$, the noise in the ciphertexts $\|\mathbf{e}_{w_j}\| \leq \beta_{i-1}(m)\delta$ and $\|\mathbf{S}_{w_j}\|_2 \leq \beta_{i-1}(m) \cdot \gamma$.
 - $\mathbf{c}_w = \mathbf{c}_{w_1} + \mathbf{c}_{w_2} = (x_{w_1}\mathbf{G} + \mathbf{B}_{w_1})^T \mathbf{s} + \mathbf{e}_{w_1} + (x_{w_2}\mathbf{G} + \mathbf{B}_{w_2})^T \mathbf{s} + \mathbf{e}_{w_2} = (x_w\mathbf{G} + \mathbf{B}_w)^T \mathbf{s} + \mathbf{e}_w$
 - $\|\mathbf{e}_w\| = \|\mathbf{e}_{w_1} + \mathbf{e}_{w_2}\| \leq \|\mathbf{e}_{w_1}\| + \|\mathbf{e}_{w_2}\| \leq (\beta_{i-1}(m) + \beta_{i-1}(m))\delta \leq \beta_i(m)\delta$
 - $\mathbf{B}_w = \mathbf{B}_{w_1} + \mathbf{B}_{w_2} = (\mathbf{A}\mathbf{S}_{w_1} - x_{w_1}\mathbf{G}) + (\mathbf{A}\mathbf{S}_{w_2} - x_{w_2}\mathbf{G}) = \mathbf{A}(\mathbf{S}_{w_1} + \mathbf{S}_{w_2}) - (x_{w_1} + x_{w_2})\mathbf{G} = \mathbf{A}\mathbf{S}_w - x_w\mathbf{G}$
 - $\|\mathbf{S}_w\|_2 = \|\mathbf{S}_{w_1} + \mathbf{S}_{w_2}\|_2 \leq \|\mathbf{S}_{w_1}\|_2 + \|\mathbf{S}_{w_2}\|_2 \leq (\beta_{i-1}(m) + \beta_{i-1}(m)) \cdot \gamma \leq \beta_i(m) \cdot \gamma$.
- Consider a gate which has input wires u and $i \in [\ell]$, output wire w and which computes multiplication. Suppose $\|\mathbf{e}_u\| \leq \beta_{i-1}(m)$ and $\|\mathbf{S}_u\|_2 \leq \beta_{i-1}(m) \cdot \gamma$, then the following holds
 - $\mathbf{c}_w = x_i\mathbf{c}_u + \mathbf{R}^T\mathbf{c}_i = x_i(x_u\mathbf{G} + \mathbf{B}_u)^T \mathbf{s} + x_i\mathbf{e}_u + \mathbf{R}^T(x_i\mathbf{G} + \mathbf{B}_i)^T \mathbf{s} + \mathbf{R}^T\mathbf{e}_i = (x_w\mathbf{G} + x_i(\mathbf{B}_u + \mathbf{G}\mathbf{R}) + \mathbf{B}_w)^T \mathbf{s} + \mathbf{e}_w$
 - $\|\mathbf{e}_w\|_2 = \|x_i\mathbf{e}_u + \mathbf{R}^T\mathbf{e}_i\|_2 \leq p\|\mathbf{e}_u\|_2 + m\|\mathbf{e}_i\|_2 \leq (p\beta_{i-1}(m) + m)\delta \leq \beta_i(m) \cdot \delta$
 - $\mathbf{B}_w = \mathbf{B}_i\mathbf{R} = (\mathbf{A}\mathbf{S}_i - x_i\mathbf{G})\mathbf{R} = \mathbf{A}\mathbf{S}_i\mathbf{R} + x_i\mathbf{B}_u = \mathbf{A}\mathbf{S}_i\mathbf{R} + x_i(\mathbf{A}\mathbf{S}_u - x_u\mathbf{G}) = \mathbf{A}(x_i\mathbf{S}_u + \mathbf{S}_i\mathbf{R}) - (x_ix_u)\mathbf{G} = \mathbf{A}\mathbf{S}_w - x_w\mathbf{G}$
 - $\|\mathbf{S}_w\|_2 = \|x_i\mathbf{S}_u + \mathbf{S}_i\mathbf{R}\|_2 \leq (p\beta_{i-1}(m) + m) \cdot \gamma \leq \beta_i(m) \cdot \gamma$.

as required. \square

Now combining Lemma 5.3 and lemmas analogous to Lemmas 4.6, 4.7 we can build an ABE system for a set of functions \mathcal{F} which can be computed by depth D circuits with (k, d, g) -complexity gates. The bound function will then be

$$\alpha_{\mathcal{F}}(n) = (\beta_d(m))^D \cdot 20\sqrt{m} = O((p^d m)^D \sqrt{m}).$$

The time complexity of the Eval algorithms for circuit C that consists of (k, d, g) -complexity gates will be $O(g \cdot |C|)$.

5.2.1 Example applications for polynomial gates

Unbounded fan-in OR gate. Assuming that boolean inputs are interpreted as integers in $\{0, 1\}$, the OR gate of ℓ inputs can be computed with the following recursive formula:

$$\text{OR}_{\ell+1}(x_1, \dots, x_\ell, x_{\ell+1}) = x_{\ell+1} + (1 - x_{\ell+1}) \cdot \text{OR}_\ell(x_1, \dots, x_\ell), \quad \text{where } \text{OR}_1(x_1) = x_1.$$

It is easy to see that OR_ℓ has restricted complexity $(\ell, 3\ell, 3\ell)$, since at each of the ℓ iterations we do one multiplication by $x_{\ell+1}$ and two fan-in 2 additions. Therefore, by Lemma 5.3, an OR_ℓ gate increases the noise in the ciphertext by a factor of $O(\ell \cdot m)$.

If we were computing the OR_ℓ function with addition and multiplication gates as in Section 4.3, the most efficient way would be to use the De Morgan's law:

$$\text{OR}_{\ell+1}(x_1, \dots, x_\ell, x_{\ell+1}) = 1 - (1 - x_1)(1 - x_2) \dots (1 - x_\ell).$$

This function can be computed with one level of ℓ fan-in-2 addition gates (to compute $(1 - x_i)$ for $i \in [\ell]$), one level of a single fan-in- ℓ multiplication gate (to compute $\prod_{i=1}^{\ell} (1 - x_i)$) and one more level of a single fan-in-2 addition gate. The noise then will grow by a factor of $O(\ell \cdot m^3)$, which will make the scheme 3 times less efficient.

The Fibonacci polynomial. Consider the following polynomial, defined for $\mathbf{x} \in [-p, p]^\ell$ using the following recurrence:

$$\begin{aligned} \Pi_1(\mathbf{x}) &= x_1, & \Pi_2(\mathbf{x}) &= x_2 \\ \Pi_{i+2}(\mathbf{x}) &= \Pi_{i+1}(\mathbf{x}) + \Pi_i(\mathbf{x}) \cdot x_{i+2} \quad \text{for } i \in \{1, \dots, \ell - 2\} \end{aligned}$$

If expanded, the number of monomials in Π_ℓ is equal to the ℓ -th Fibonacci number, which is exponential in ℓ . The degree of the polynomial is $\lfloor \frac{\ell}{2} \rfloor$. The recurrence shows that the restricted arithmetic complexity of this polynomial is $(\ell, \ell, 2\ell)$. Therefore, we can compute it with a single polynomial gate and, by Lemma 5.3, the growth in ciphertext noise will be proportional to $p^\ell \cdot m$.

We conjecture that computing this polynomial with a polynomial-size arithmetic circuit requires linear depth in ℓ . Therefore, the growth in ciphertext noise using the approach of Section 4.3 will be proportional to $(pm)^{O(\ell)}$ which is much worse.

6 ABE with Short Ciphertexts from Multi-linear Maps

We assume familiarity with multi-linear maps [BS02, GGH13a], which we overview in Section 2.3.

Intuition. We assume that the circuits consist of AND and OR gates. To handle general circuits (with negations), we can apply De Morgan's rule to transform it into a monotone circuit, doubling the number of input attributes (similar to [GGH⁺13c]).

The inspiration of our construction comes from the beautiful work of Applebaum, Ishai, Kushilevitz and Waters [AIKW13] who show a way to compress the garbled input in a (single use) garbling scheme all the way down to size $|\mathbf{x}| + \text{poly}(\lambda)$. This is useful to us in the context of ABE schemes due to a simple connection between ABE and *reusable* garbled circuits with authenticity observed in [GVW13]. In essence, they observe that the secret key for a function f in an ABE scheme corresponds to the garbled circuit for f , and the ciphertext encrypting an attribute vector \mathbf{x} corresponds

to the garbled input for \mathbf{x} in the reusable garbling scheme. Thus, the problem of compressing ciphertexts down to size $|\mathbf{x}| + \text{poly}(\lambda)$ boils down to the question of generalizing [AIKW13] to the setting of *reusable* garbling schemes. We are able to achieve this using multilinear maps.

Security of the scheme relies on a generalization of the bilinear Diffie-Hellman Exponent Assumption to the multi-linear setting (see Definition 2.9).¹ The bilinear Diffie-Hellman Exponent Assumption was recently used to prove the security of the first broadcast encryption with constant size ciphertexts [BGW05] (which in turn can be thought of as a special case of ABE with short ciphertexts.)

Theorem 6.1 (Selective security). *For all polynomials $d_{\max} = d_{\max}(\lambda)$, there exists a selectively-secure attribute-based encryption with ciphertext size $\text{poly}(d_{\max})$ for any family of polynomial-size circuits with depth at most d_{\max} and input size ℓ , assuming hardness of $(d+1, \ell)$ -Multilinear Diffie-Hellman Exponent Assumption.*

6.1 Our Construction

- **Params**($1^\lambda, d_{\max}$): The parameters generation algorithm takes the security parameter and the maximum circuit depth. It generates a multi-linear map $\mathcal{G}(1^\lambda, k = d+1)$ that produces groups (G_1, \dots, G_k) along with a set of generators g_1, \dots, g_k and map descriptors $\{e_{ij}\}$. It outputs the public parameters $pp = (\{G_i, g_i\}_{i \in [k]}, \{e_{ij}\}_{i,j \in [k]})$, which are implicitly known to all of the algorithms below.
- **Setup**(1^ℓ): For each input bit $i \in \{1, 2, \dots, \ell\}$, choose a random element q_i in \mathbb{Z}_p . Let $g = g_1$ be the generator of the first group. Define $h_i = g^{q_i}$. Also, choose α at random from \mathbb{Z}_p and let $t = g^\alpha$. Set the master public key

$$\text{mpk} := (h_1, \dots, h_\ell, t)$$

and the master secret key as $\text{msk} := \alpha$.

- **Keygen**(msk, C): The key-generation algorithm takes a circuit C with ℓ input bits and a master secret key msk and outputs a secret key sk_C defined as follows.
 1. Choose randomly $((r_1, z_1), \dots, (r_\ell, z_\ell))$ from \mathbb{Z}_q^2 for each input wire of the circuit C . In addition, choose $((r_{\ell+1}, a_{\ell+1}, b_{\ell+1}), \dots, (r_n, a_n, b_n))$ from \mathbb{Z}_q^3 randomly for all internal wires of C .
 2. Compute an $\ell \times \ell$ matrix \tilde{M} , where all diagonal entries (i, i) are of the form $(h_i)^{z_i} g^{r_i}$ and all non-diagonal entries (i, j) are of the form $(h_i)^{z_j}$. Append g^{-z_i} as the last row of the matrix and call the resulting matrix M .
 3. Consider a gate $\Gamma = (u, v, w)$ where wires u, v are at depth $j-1$ and w is at depth j . If Γ is an OR gate, compute

$$K_\Gamma = (K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u}, K_\Gamma^4 = g_j^{r_w - b_w r_v})$$

Else if Γ is an AND gate, compute

$$K_\Gamma = (K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u - b_w r_v})$$

¹Our construction can be converted to multi-linear graded-encodings, recently instantiated by Garg et al. [GGH13a] and Coron et al. [CLT13].

4. Set $\sigma = g_{k-1}^{\alpha-r_n}$
5. Define and output the secret key as

$$\mathbf{sk}_C := (C, \{K_\Gamma\}_{\Gamma \in C}, M, \sigma)$$

- **Enc(mpk, \mathbf{x}, μ):** The encryption algorithm takes the master public key \mathbf{mpk} , an index $\mathbf{x} \in \{0, 1\}^\ell$ and a message $\mu \in \{0, 1\}$, and outputs a ciphertext $\mathbf{c}_\mathbf{x}$ defined as follows. Choose a random element s in \mathbb{Z}_q . Let X be the set of indices i such that $x_i = 1$. Let $\gamma_0 = t^s$ if $\mu = 1$, otherwise let γ_0 be a randomly chosen element from G_k . Output ciphertext as

$$\mathbf{c}_\mathbf{x} := \left(\mathbf{x}, \gamma_0, g^s, \gamma_1 = \left(\prod_{i \in X} h_i \right)^s \right)$$

- **Dec($\mathbf{sk}_C, \mathbf{c}_\mathbf{x}$):** The decryption algorithm takes the ciphertext $\mathbf{c}_\mathbf{x}$, and secret key \mathbf{sk}_C and proceeds as follows. If $C(\mathbf{x}) = 0$, it outputs \perp . Otherwise,

1. Let X be the set of indices i such that $x_i = 1$. For each input wire $i \in X$, using the matrix M compute $g^{r_i} \left(\prod_{j \in X} h_j \right)^{z_i}$ and then

$$\begin{aligned} g_2^{r_i s} &= e \left(g^s, g^{r_i} \left(\prod_{j \in X} h_j \right)^{z_i} \right) \cdot e \left(\gamma_1, g^{-z_i} \right) \\ &= e \left(g^s, g^{r_i} \left(\prod_{j \in X} h_j \right)^{z_i} \right) \cdot e \left(\left(\prod_{j \in X} h_j \right)^s, g^{-z_i} \right) \end{aligned}$$

2. Now, for each gate $\Gamma = (u, v, w)$ where w is a wire at level j , (recursively going from the input to the output) compute $g_{j+1}^{r_w s}$ as follows:
 - If Γ is an OR gate, and $C(\mathbf{x})_u = 1$, compute $g_{j+1}^{r_w s} = e(K_\Gamma^1, g_j^{r_u s}) \cdot e(g^s, K_\Gamma^3)$.
 - Else if $C(\mathbf{x})_v = 1$, compute $g_{j+1}^{r_w s} = e(K_\Gamma^2, g_j^{r_v s}) \cdot e(g^s, K_\Gamma^4)$.
 - Else if Γ is an AND gate, compute $g_{j+1}^{r_w s} = e(K_\Gamma^1, g_j^{r_u s}) \cdot e(K_\Gamma^2, g_j^{r_v s}) \cdot e(g^s, K_\Gamma^3)$.

3. If $C(\mathbf{x}) = 1$, then the user computes $g_k^{r_n s}$ for the output wire. Finally, compute

$$\psi = e(g^s, \sigma) \cdot g_k^{r_n s} = e(g^s, g_{k-1}^{\alpha-r_n}) \cdot g_k^{r_n s}$$

4. Output $\mu = 1$ if $\psi = \gamma_0$, otherwise output 0.

6.2 Correctness

Claim 6.2. For all active wires w at level j (that is, $C(\mathbf{x})_w = 1$) the user holds $g_{i+1}^{sr_w}$.

Proof. Clearly, the base case is satisfied as shown above. Now consider a gate $\Gamma = (u, v, w)$. If g is an OR gate and assume $C(x)_u = 1$, then

$$\begin{aligned} g_{j+1}^{sr_w} &= e(K_\Gamma^1, g_j^{r_u s}) \cdot e(g^s, K_\Gamma^3) \\ &= e(g^{a_w}, g_j^{r_u s}) \cdot e(g^s, g_j^{r_w - a_w r_u}) \\ &= e(g, g_j)^{a_w r_u s} \cdot e(g, g_j)^{sr_w} \cdot e(g, g_j)^{-a_w r_u s} \end{aligned}$$

The case when $C(x)_v = 1$ is similar. Also, if g is an AND gate, then

$$\begin{aligned}
g_{j+1}^{sr_w} &= e(K_\Gamma^1, g_j^{r_{us}}) \cdot e(K_\Gamma^2, g_j^{r_{vs}}) \cdot e(g^s, K_\Gamma^3) \\
&= e(g^{a_w}, g_j^{r_{us}}) \cdot e(g^{b_w}, g_j^{r_{vs}}) \cdot e(g^s, g_j^{r_w - a_w r_u - b_w r_v}) \\
&= e(g, g_j)^{a_w r_{us}} \cdot e(g, g_j)^{b_w r_{vs}} \cdot e(g, g_j)^{sr_w} \cdot e(g, g_j)^{-a_w r_{us} - b_w r_{vs}} \\
&= e(g, g_j)^{a_w r_{us} + b_w r_{vs}} \cdot e(g, g_j)^{sr_w} \cdot e(g, g_j)^{-a_w r_{us} - b_w r_{vs}}
\end{aligned}$$

Hence, if $C(x) = 1$, the user computes $g_k^{sr_n}$ and so

$$\begin{aligned}
\psi &= e(g^s, \sigma) \cdot g_k^{r_n s} \\
&= e(g^s, g_{k-1}^{\alpha - r_n}) \cdot g_k^{r_n s} \\
&= g_k^{\alpha s} = t^s = \gamma_0
\end{aligned}$$

if $m = 1$. □

6.3 Security Proof

Assume there is an adversary Adv^* that breaks the security of the ABE scheme. We construct an adversary Adv that breaks the (k, ℓ) -Multi-linear Diffie-Hellman Exponent Assumption. The adversary Adv is given a challenge

$$(g^{c_1}, \dots, g^{c_1^\ell}, \dots, g^{c_1^{\ell+2}}, \dots, g^{c_1^{2\ell}}, g^{c_2}, \dots, g^{c_k}, \beta)$$

where β is either $g_k^{c_1^{\ell+1} \prod_{2 \leq i \leq k} c_i}$ or a random element of G_k . The adversary invokes Adv^* and gets x^* as the challenge index. Let X be the set of indices i such that $x_i = 1$. The adversary will ensure the following induction: for every inactive wire w at depth j , $r_w = c_1^{\ell+1} \prod_{2 \leq i \leq j} c_i$ (plus known randomness). Hence, for all input wires w , $r_w = c_1^{\ell+1}$ (plus known randomness).

We now define simulated experiments which Adv will be using to break the assumption.

- **Setup $^*(1^\ell)$:** For each input bit $i \notin X$, choose a random element b_i in \mathbb{Z}_q and implicitly set $q_i = c_1^{\ell+1-i} + b_i$. For each $i \in X$, choose a random $q_i \in \mathbb{Z}_q$. Let $g = g_1$ be the generator of the first group. For all i , compute $h_i = g^{q_i}$. Randomly choose γ and let $t = g_k^\alpha = g_k^{c_1^{\ell+1} \prod_{2 \leq i \leq k-1} c_i + \gamma}$ which can be computed from the challenge component by repeated pairing. Set the master public key

$$\text{mpk} := (h_1, \dots, h_\ell, t)$$

and the master secret key as $\text{msk} := \perp$.

- **Keygen $^*(C, \text{msk})$:** The key-generation algorithm takes a circuit C with ℓ input bits and a master secret key msk and outputs a secret key sk_C defined as follows.
 1. For all $i \in X$, choose randomly $r_i \in \mathbb{Z}_q$. For all $i \notin X$, randomly choose $f_i \in \mathbb{Z}_q$ and implicitly set $r_i = c_1^{\ell+1} + f_i$ (that is, we embed the challenge into the attributes $\notin X$).
 2. For all $i \in [\ell]$, choose $p_i \in \mathbb{Z}_q$ at random and implicitly set $z_i = -c_1^i + p_i$.

3. Compute the matrix M :

$$M := \begin{bmatrix} g^{-z_1} & g^{-z_2} & g^{-z_3} & \dots & g^{-z_\ell} \\ (h_1)^{z_1} g^{r_1} & (h_1)^{z_2} & (h_1)^{z_3} & \dots & (h_1)^{z_\ell} \\ (h_2)^{z_1} & (h_2)^{z_2} g^{r_2} & (h_2)^{z_3} & \dots & (h_2)^{z_\ell} \\ (h_3)^{z_1} & (h_3)^{z_2} & (h_3)^{z_3} g^{r_3} & \dots & (h_3)^{z_\ell} \\ \vdots & & \ddots & & \vdots \\ (h_\ell)^{z_1} & (h_\ell)^{z_2} & (h_\ell)^{z_3} & \dots & (h_\ell)^{z_\ell} g^{r_\ell} \end{bmatrix}$$

4. We now argue that the adversary can compute every entry in the matrix M .

- (a) Entries of the first row can be computed by $g^{-z_i} = g^{c_1^i - p_i} = g^{c_1^i} \cdot g^{-p_i}$, where p_i is known.
- (b) Note that for all $i = j$ (i.e. the diagonal entries). If $i \notin X$, then

$$(h_i)^{z_i} \cdot g^{r_i} = g^{(c_1^{\ell+1-i} + b_i)(-c_1^i + p_i)} \cdot g^{c_1^{\ell+1} + f_i} = g^{c_1^{\ell+1-i} p_i - b_i c_1^i + b_i p_i + f_i}$$

If $i \in X$, then q_i, z_i, r_i are all known.

- (c) Now, consider non-diagonal entries $i \neq j$. If $i \notin X$ and $j \in X$, then

$$(h_i)^{z_j} = (g^{c_1^{\ell+1-i} + b_i})^{-c_1^j + p_j} = g^{-c_1^{\ell+1-i} + j} \cdot g^{-b_i c_1^j} \cdot g^{p_j c_1^{\ell+1-i}} \cdot g^{b_i p_j}$$

which can be computed given the challenge and the knowledge of b_i, p_j . Also, if $i \in X$ and $j \notin X$, similarly

$$(h_i)^{z_j} = (g^{q_i})^{-c_1^j + p_j} = g^{-c_1^j q_i} \cdot g^{q_i p_j}$$

can be computed given the challenge and the knowledge of q_i, p_j .

5. Consider a gate $\Gamma = (u, v, w)$ where wires u, v are at depth $j - 1$ and w is at depth j .

- (a) If Γ is an OR gate and $C(x^*)_w = 1$, then values r_w, a_w, b_w are randomly chosen from \mathbb{Z}_q . Otherwise, we implicitly set $a_w = c_j + d_w, b_w = c_j + k_w$, where $d_w, k_w \in \mathbb{Z}_q$ are randomly chosen and c_j is the value a part of the challenge. Also, implicitly set $r_w = c_1^{\ell+1} \prod_{2 \leq i \leq j} c_i + e_w$, where $e_w \in \mathbb{Z}_q$ is randomly chosen. Compute

$$K_\Gamma = (K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u}, K_\Gamma^4 = g_j^{r_w - b_w r_v})$$

Note that in the case $C(x^*)_w = 0$,

$$\begin{aligned} r_w - a_w r_u &= c_1^{\ell+1} \prod_{2 \leq i \leq j} c_i + e_w - (c_j + d_w)(c_1^{\ell+1} \prod_{2 \leq i \leq j-1} c_i + n_u) \\ &= -c_j n_u - d_w (c_1^{\ell+1} \prod_{2 \leq i \leq j-1} c_i) - d_w n_u + e_w \end{aligned}$$

Hence, component K_Γ^3 can be computed by pairing j elements from the challenge: $g^{c_1}, g^\ell, g^{c_2}, \dots, g^{c_{j-1}}$. Similarly, for term K_Γ^4 .

- (b) Else if Γ is an AND gate and $C(x^*)_w = 1$, then values r_w, a_w, b_w are randomly chosen from \mathbb{Z}_q . And the adversary computes

$$K_\Gamma = (K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u - b_w r_v})$$

Otherwise, if $C(x^*)_u = 0$, then implicitly set $r_w = c_1^{\ell+1} \prod_{2 \leq i \leq j} c_i + e_w$, $a_w = c_j + d_w$ where e_w, d_w are randomly chosen. Also, choose b_w at random. Again, the adversary can compute

$$K_\Gamma = (K_\Gamma^1 = g^{a_w}, K_\Gamma^2 = g^{b_w}, K_\Gamma^3 = g_j^{r_w - a_w r_u - b_w r_v})$$

Note that,

$$\begin{aligned} r_w - a_w r_u - b_w r_v &= c_1^{\ell+1} \prod_{2 \leq i \leq j} c_i + e_w - (c_j + d_w) (c_1^{\ell+1} \prod_{2 \leq i \leq j-1} c_i + n_u) - b_w r_v \\ &= e_w - c_j n_u - d_w (c_1^{\ell+1} \prod_{2 \leq i \leq j-1} c_i) - d_w n_u - b_w r_v \end{aligned}$$

Hence, K_Γ^3 can be computed by the adversary by applying j pairings to the challenge components $g^{c_1}, g^\ell, g^{c_2}, \dots, g^{c_{j-1}}$ and using the other *known* randomness components.

The adversary performs the symmetric operations if $C(x^*)_v = 0$.

6. Set $\sigma = g_{k-1}^{\alpha - r_n}$. Note that since $C(x^*) = 0$ the component r_n embeds parts challenge into it. Hence, σ can be computed by the adversary due to cancellation in the exponent:

$$\alpha - r_n = c_1^{\ell+1} \prod_{2 \leq i \leq k-1} c_i + \gamma - c_1^{\ell+1} \prod_{2 \leq i \leq k-1} c_i + e_n = \gamma + e_n$$

7. Define and output the secret key as

$$\text{sk}_C := (C, \{K_\Gamma\}_{g \in C}, \sigma)$$

- $\text{Enc}^*(\text{mpk}, x^*, m)$: The encryption algorithm takes the master public key mpk , an index x^* and a message m , and outputs a ciphertext ct_{x^*} defined as follows. Let X be the set of indices i such that $x_i^* = i$. Implicitly let $s = c_k$. Let $\gamma_0 = \gamma = \beta \cdot g_k^{\gamma c_k}$. Output ciphertext as

$$\text{ct}_x := \left(x, \gamma_0, g^{c_k}, \gamma_1 = \left(\prod_{i \in X} h_i \right)^{c_k} \right)$$

where b is a randomly chosen bit. Note that $(\prod_{i \in X} h_i)^s$ can be computed given the challenge component g^{c_k} and known randomness q_i for $i \in X$. If $\beta = g_k^{c_1^{\ell+1} \prod_{2 \leq i \leq k} c_i}$, then,

$$\begin{aligned} \beta \cdot g_k^{\gamma c_k} &= (g_k^{c_1^{\ell+1} \prod_{2 \leq i \leq k-1} c_i} \cdot g_k^\gamma)^{c_k} \\ &= (g_k^{c_1^{\ell+1} \prod_{2 \leq i \leq k-1} c_i + \gamma})^{c_k} \\ &= t^{c_k} = t^s \end{aligned}$$

which corresponds to an encryption of 1. Otherwise, if β is a randomly chosen in G_k , this corresponds to an encryption of 0.

The adversary Adv uses the above simulated algorithms to answer the queries to Adv^* . If Adv^* returns $m = 1$, then Adv outputs that $\beta = g_k^{c_1^{\ell+1} \prod_{2 \leq i \leq k} c_i}$. Otherwise, it outputs that β is randomly chosen in the target.

7 Applications and Extensions

7.1 Single-Key Functional Encryption and Reusable Garbled Circuits

Goldwasser, Kalai, Popa, Vaikuntanathan and Zeldovich showed how to obtain a Single-Key Functional Encryption (SKFE) and Reusable Garbled Circuits from: (1) Attribute-based Encryption, (2) Fully-Homomorphic Encryption and (3) “one-time” Garbled Circuits [GKP⁺13b]. In this section we show what we gain in efficiency in the secret key and ciphertext sizes for these two construction by using our ABE schemes.

Theorem 7.1 ([GKP⁺13b]). *There is a (fully/selectively secure) single-key functional encryption scheme \mathcal{FE} for any class of circuits \mathcal{C} that take ℓ bits of input and produce a one-bit output, assuming the existence of (1) \mathcal{C} -homomorphic encryption scheme, (2) a (fully/selectively) secure ABE scheme for a related class of predicates and (3) Yao’s Garbling Scheme, where:*

1. *The size of the secret key is $2 \cdot \alpha \cdot \text{abe.keysize}$, where abe.keysize is the size of the ABE key for circuit performing homomorphic evaluation of C and outputting a bit of the resulting ciphertext.*
2. *The size of the ciphertext is $2 \cdot \alpha \cdot \text{abe.ctime}(\ell \cdot \alpha + \gamma) + \text{poly}(\lambda, \alpha, \beta)$*

where (α, β, γ) denote the sizes of the FHE (ciphertext, secret key, public key), respectively. abe.keysize , $\text{abe.ctime}(k)$ are the size of ABE secret key, ciphertext on k -bit attribute vector and λ is the security parameter.

Since FHE (and Yao’s Garbled Circuits) can also be instantiated assuming the sub-exponential hardness of LWE ([BV11], [BGV12]), we obtain the following corollaries.

Corollary 7.2. *Combining our short secret key ABE construction (Theorem-4.4) and Theorem-7.1, we obtain a single-key functional encryption scheme for a circuit class \mathcal{C} with depth at most d_{\max} , where the secret key size is some $\text{poly}(d_{\max}, \lambda)$ and λ is the security parameter.*

To obtain a short ciphertext for functional encryption scheme, we need another observation. There exists a fully-homomorphic encryption scheme where ciphertext encrypting k bits of input is of size $k + \text{poly}(\lambda)$, where λ is the security parameter. We refer the reader to the full version for further details.

Corollary 7.3. *Combining the above observation, our short ciphertext ABE construction (Theorem-6.1) and Theorem-7.1, we obtain a single-key functional encryption scheme for any circuit class \mathcal{C} with depth at most d_{\max} and ℓ bit inputs, where the size of the ciphertext is $\ell + \text{poly}(d_{\max}, \lambda)$ and λ is the security parameter.*

Next, we apply our results to get the optimal construction of reusable garbled circuits.

Theorem 7.4 ([GKP⁺13b]). *There exists a reusable garbling scheme for any class of circuits \mathcal{C} that take ℓ bits of input, assuming the existence (1) symmetric-encryption algorithm, (2) a single-key functional encryption for \mathcal{C} , where:*

1. *The size of the secret key is $|\mathcal{C}| + \text{fe.keysize} + \text{poly}(\lambda)$, where fe.keysize is the size of the FE key for circuit performing symmetric-key decryption and evaluation of \mathcal{C} .*
2. *The size of the ciphertext is $\text{fe.ctime}(\lambda + \ell)$*

where $\text{fe.ctime}(\lambda + \ell)$ is the size of FE ciphertext on $\lambda + \ell$ -bit input.

Corollary 7.5. *From Corollary-7.2 and Theorem-7.4, we obtain a reusable garbled circuits scheme for any class of polynomial-size circuits with depth at most d_{\max} , where the secret key size is $|\mathcal{C}| + \text{poly}(d_{\max}, \lambda)$.*

Corollary 7.6. *From Corollary-7.3 and Theorem-7.4, we obtain a reusable garbled circuits scheme for any class of polynomial-size circuits with depth at most d_{\max} and ℓ bit inputs, where the ciphertext size is $\ell + \text{poly}(d_{\max}, \lambda)$.*

8 Conclusions and open problems

We presented an ABE for arithmetic circuits with short secret keys whose security is based on the LWE problem. At the heart of our construction is a method for transforming a noisy vector of the form $\mathbf{c} = (\mathbf{A}|x_1\mathbf{G} + \mathbf{B}_1| \cdots |x_\ell\mathbf{G} + \mathbf{B}_\ell)^\top \mathbf{s} + \mathbf{e}$ into a vector $(\mathbf{A}|y\mathbf{G} + \mathbf{B}_f)^\top \mathbf{s} + \mathbf{e}_f$ where $y = f(x_1, \dots, x_\ell)$ and \mathbf{e}_f is not much longer than \mathbf{e} . The short decryption key sk_f provides a way to decrypt when $y = 0$. We refer to this property as a *public-key homomorphism* and expect it to find other applications.

Natural open problems that remain are a way to provide adaptive security from LWE with a polynomial-time reduction. It would also be useful to construct an efficient ABE for arithmetic circuits where multiplication gates can handle inputs as large as the modulus q .

Acknowledgments. We thank Chris Peikert for his helpful comments and for suggesting Remark 4.3.

D. Boneh is supported by NSF, the DARPA PROCEED program, an AFOSR MURI award, a grant from ONR, an IARPA project provided via DoI/NBC, and Google faculty award. Opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA or IARPA.

S. Gorbunov is supported by Alexander Graham Bell Canada Graduate Scholarship (CGSD3).

G. Segev is supported by the European Union's Seventh Framework Programme (FP7) via a Marie Curie Career Integration Grant, by the Israel Science Foundation (Grant No. 483/13), and by the Israeli Centers of Research Excellence (I-CORE) Program (Center No. 4/11).

V. Vaikuntanathan is supported by an NSERC Discovery Grant, DARPA Grant number FA8750-11-2-0225, a Connaught New Researcher Award, an Alfred P. Sloan Research Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.

References

- [ABB10] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (H)IBE in the standard model. In *EUROCRYPT*, 2010.
- [ABV⁺12] S. Agrawal, X. Boyen, V. Vaikuntanathan, P. Voulgaris, and H. Wee. Functional encryption for threshold functions (or fuzzy ibe) from lattices. In *PKC*, 2012.
- [AFV11] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, 2011.
- [AIKW13] B. Applebaum, Y. Ishai, E. Kushilevitz, and B. Waters. Encoding functions with constant online rate or how to compress garbled circuits keys. In *CRYPTO*, 2013.
- [Ajt99] M. Ajtai. Generating hard instances of the short basis problem. In *ICALP*, 1999.
- [ALdP11] N. Attrapadung, B. Libert, and E. de Panafieu. Expressive key-policy attribute-based encryption with constant-size ciphertexts. In *Public Key Cryptography*, volume 6571, pages 90–108, 2011.
- [AP09] J. Alwen and C. Peikert. Generating shorter bases for hard random lattices. In *STACS*, 2009.
- [BB11] D. Boneh and X. Boyen. Efficient selective identity-based encryption without random oracles. *Journal of Cryptology*, 24(4):659–693, 2011.
- [BF03] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003. Preliminary version in *CRYPTO* ’01.
- [BGG⁺14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit abe, and compact garbled circuits. In *Proc. of Eurocrypt’14*, 2014.
- [BGV12] Z. Brakerski, C. Gentry, and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, 2012.
- [BGW05] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, 2005.
- [BHHI10] B. Barak, I. Haitner, D. Hofheinz, and Y. Ishai. Bounded key-dependent message security. In *EUROCRYPT*, 2010.
- [BMR90] D. Beaver, S. Micali, and P. Rogaway. The round complexity of secure protocols (extended abstract). In *STOC*, 1990.
- [BNS13] Dan Boneh, Valeria Nikolaenko, and Gil Segev. Attribute-based encryption for arithmetic circuits. Cryptology ePrint Archive, Report 2013/669, 2013. <http://eprint.iacr.org/>.
- [Boy13] X. Boyen. Attribute-based functional encryption on lattices. In *TCC*, 2013.

- [BS02] D. Boneh and A. Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324:71–90, 2002.
- [BSW11] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.
- [BV11] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, 2011.
- [BW07] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.
- [BW13] D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, 2013.
- [CHKP10] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, 2010.
- [CLT13] J. Coron, T. Lepoint, and M. Tibouchi. Practical multilinear maps over the integers. In *CRYPTO*, 2013.
- [Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, 2001.
- [FN93] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO*, 1993.
- [GGH13a] S. Garg, C. Gentry, and S. Halevi. Candidate multilinear maps from ideal lattices. In *EUROCRYPT*, 2013.
- [GGH⁺13b] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.
- [GGH⁺13c] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.
- [GGH⁺13d] Craig Gentry, Sergey Gorbunov, Shai Halevi, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. How to compress (reusable) garbled circuits. Cryptology ePrint Archive, Report 2013/687, 2013. <http://eprint.iacr.org/>.
- [GGP10] R. Gennaro, C. Gentry, and B. Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, 2010.
- [GGSW13] S. Garg, C. Gentry, A. Sahai, and B. Waters. Witness encryption and its applications. In *STOC*, 2013.
- [GHV10] C. Gentry, S. Halevi, and V. Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, 2010.
- [GKP⁺13a] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *CRYPTO*, 2013.

- [GKP⁺13b] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.
- [GKR08] S. Goldwasser, Y. T. Kalai, and G. N. Rothblum. Delegating computation: interactive proofs for muggles. In *STOC*, 2008.
- [GMW87] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, 1987.
- [GPSW06] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, 2006.
- [GPV08] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, 2008.
- [GVW13] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *STOC*, 2013.
- [HW13] S. Hohenberger and B. Waters. Attribute-based encryption with fast decryption. In *PKC*, 2013.
- [KS08] V. Kolesnikov and T. Schneider. Improved garbled circuit: Free xor gates and applications. In *ICALP*, 2008.
- [KSW08] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.
- [LO13] S. Lu and R. Ostrovsky. How to garble ram programs. In *EUROCRYPT*, 2013.
- [LOS⁺10] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.
- [LPRTJ05] A. Litvak, A. Pajor, M. Rudelson, and N. Tomczak-Jaegermann. Smallest singular value of random matrices and geometry of random polytopes. *Advances in Mathematics*, 195(2):491–523, 2005.
- [LW12] A. B. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, 2012.
- [MP12] D. Micciancio and C. Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *EUROCRYPT*, 2012.
- [OT10] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, 2010.
- [Pei09] C. Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, 2009.
- [PRV12] B. Parno, M. Raykova, and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC*, 2012.

- [PTMW06] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *ACM CCS*, 2006.
- [Reg05] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, 2005.
- [Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, 1984.
- [Sho08] Victor Shoup. *A Computational Introduction to Number Theory and Algebra, second edition*. Cambridge University Press, 2008.
- [SW05] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.
- [Wat12] B. Waters. Functional encryption for regular languages. In *CRYPTO*, 2012.
- [Yao86] A. C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, 1986.

Reusable Garbled Circuits and Succinct Functional Encryption

Shafi Goldwasser[★] Yael Kalai[†] Raluca Ada Popa[★]
Vinod Vaikuntanathan[⊠] Nickolai Zeldovich[★]

★ MIT CSAIL † Microsoft Research ⊠ University of Toronto

March 24, 2013

Abstract

Garbled circuits, introduced by Yao in the mid 80s, allow computing a function f on an input x without leaking anything about f or x besides $f(x)$. Garbled circuits found numerous applications, but every known construction suffers from one limitation: it offers no security if used on multiple inputs x . In this paper, we construct for the first time *reusable* garbled circuits. The key building block is a new *succinct single-key functional encryption* scheme.

Functional encryption is an ambitious primitive: given an encryption $\text{Enc}(x)$ of a value x , and a secret key sk_f for a function f , anyone can compute $f(x)$ without learning any other information about x . We construct, for the first time, a *succinct* functional encryption scheme for *any* polynomial-time function f where succinctness means that the ciphertext size does not grow with the size of the circuit for f , but only with its depth. The security of our construction is based on the intractability of the Learning with Errors (LWE) problem and holds as long as an adversary has access to a *single key* sk_f (or even an a priori bounded number of keys for different functions).

Building on our succinct single-key functional encryption scheme, we show several new applications in addition to reusable garbled circuits, such as a paradigm for general function obfuscation which we call token-based obfuscation, homomorphic encryption for a class of Turing machines where the evaluation runs in input-specific time rather than worst-case time, and a scheme for delegating computation which is publicly verifiable and maintains the privacy of the computation.

Contents

1	Introduction	3
1.1	Our Results	4
1.1.1	Main Application: Reusable Garbled Circuits	6
1.1.2	Token-Based Obfuscation: a New Way to Circumvent Obfuscation Impossibility Results	6
1.1.3	Computing on Encrypted Data in Input-Specific Time	7
1.1.4	Publicly Verifiable Delegation with Secrecy	8
1.2	Technique Outline	8
2	Preliminaries	11
2.1	Notation	11
2.2	Background on Learning With Errors (LWE)	12
2.3	Fully Homomorphic Encryption (FHE)	12
2.4	Background on Garbled Circuits	13
2.5	Attribute-Based Encryption (ABE)	15
2.5.1	Two-Outcome Attribute-Based Encryption	17
2.6	Functional Encryption (FE)	18
2.6.1	Security of Functional Encryption	19
3	Our Functional Encryption Scheme	20
3.1	Construction	23
3.2	Proof	24
4	Reusable Garbled Circuits	29
4.1	Construction	31
4.2	Proof	32
4.3	Impossibility of Public-Key Reusable Garbled Circuits	35
5	Token-Based Obfuscation	36
5.1	Definition	36
5.2	Scheme	37
6	Computing on Encrypted Data in Input-Specific Time	38
6.1	Construction	40
6.2	Results	41
6.3	Input-Dependent Output Size	43
A	Detailed Background on Learning With Errors (LWE)	46
B	Construction of Two-Outcome Attribute-Based Encryption	47
C	Homomorphic Encryption for Turing Machines: Definitions and Proofs	49
C.1	Proof	50

1 Introduction

Breaches of confidential data are commonplace: personal information of millions of people, such as financial, medical, customer, and employee data, is disclosed every year [Pri12, Ver]. These disclosures often happen because untrustworthy systems handle confidential data. As applications move to cloud computing platforms, ensuring data confidentiality on third-party servers that may be untrustworthy becomes a top concern [Dav12].

A powerful technique for preventing data disclosures without having to ensure the server is trustworthy is to encrypt the data provided to the server and then compute on the encrypted data. Thus, if the server does not have access to the plaintext or to the decryption key, it will be unable to disclose confidential data. The big leap of the last decade towards computing over encrypted data has been fully homomorphic encryption (FHE) [Gen09, DGHV10, SS10b, BV11b, BV11a, Vai11, BGV12, GHS12a, GHS12b, LTV12, Bra12].

A fundamental question with this approach is: *who can decrypt the results of computations on encrypted data?* If data is encrypted using FHE, anyone can perform a computation on it (with knowledge of the public key), while the result of the computation can be decrypted only using the secret key. However, the secret key allows decrypting *all* data encrypted under the corresponding public key. This model suffices for certain applications, but it rules out a large class of applications in which the party computing on the encrypted data needs to determine the computation result on its own. For example, spam filters should be able to determine if an encrypted email is spam and discard it, without learning anything else about the email's content. With FHE, the spam filter can run the spam detection algorithm homomorphically on an encrypted email and obtain an encrypted result; however, it cannot tell if the algorithm deems the email spam or not. Having the data owner provide the decryption key to the spam filter is not a solution: the spam filter can now decrypt all the emails as well!

A promising approach to this problem is *functional encryption* [SW05, GPSW06, KSW08, LOS⁺10, OT10, O'N10, BSW]. In functional encryption, anyone can encrypt data with a master public key mpk and the holder of the master secret key can provide keys for functions, for example sk_f for function f . Anyone with access to a key sk_f and a ciphertext c for x can obtain the result of the computation in plaintext form: $f(x)$. The security of FE requires that the adversary does not learn anything about x , other than the computation result $f(x)$. It is easy to see, for example, how to solve the above spam filter problem with a functional encryption scheme. A user Alice publishes her public key online and gives the spam filter a key for the filtering function. Users sending email to Alice will encrypt the email with her public key. The spam filter can now determine by itself, for each email, whether to store it in Alice's mailbox or to discard it as spam, without learning anything about Alice's email (except for whether it was deemed spam or not).

The recent impossibility result of Agrawal, Gorbunov, Vaikuntanathan and Wee [AGVW12] says that functional encryption schemes where an adversary can receive an arbitrary number of keys for general functions are impossible for a natural simulation-based security definition;¹ stated differently, any functional encryption scheme that can securely provide q keys for general functions must have ciphertexts growing linearly in q . Since any scheme that can securely provide a single key yields a scheme that can securely provide q keys by repetition, the question becomes if one can construct a functional encryption scheme that can securely provide a *single key for a general function* under this simulation-based security definition. Such a single-key functional encryption scheme is a powerful tool, enabling the applications we will discuss.

In this paper, we construct the first single-key functional encryption scheme for a *general* function that is *succinct*: the size of the ciphertext grows with the depth d of the circuit computing the function and is

¹This impossibility result holds for non-adaptive simulation-based security, which is weaker than some existing simulation-based definitions such as adaptive security. Nevertheless, this result does not carry over to indistinguishability-based definitions, for which possibility or impossibility is currently an open question. In this paper, we are interested in achieving the simulation-based definition.

independent of the size of the circuit. Up until our work, the known constructions of functional encryption were quite limited. First, the works of Boneh and Waters [BW07], Katz, Sahai and Waters [KSW08], Agrawal, Freeman and Vaikuntanathan [AFV11], and Shen, Shi and Waters [SSW09] show functional encryption schemes (based on different assumptions) for a very simple function: the inner product function f_y (or a variant of it), that on input x outputs 1 if and only if $\langle x, y \rangle = 0$.² These works do not shed light on how to extend beyond inner products. Second, Sahai and Seyalioglu [SS10a] and Gorbunov, Vaikuntanathan and Wee [GVW12] provide a construction for single-key functional encryption for one general function with a *non-succinct* ciphertext size (at least the size of a universal circuit computing the functions allowed by the scheme³). [SS10a] was the first to introduce the idea of single-key functional encryption and [GVW12] also extends it to allow the adversary to see secret keys for q functions of his choice, by increasing the size of the ciphertexts linearly with q where q is known in advance.⁴ We emphasize that the non-succinctness of these schemes is particularly undesirable and it precludes many useful applications of functional encryption (e.g., delegation, reusable garbled circuits, FHE for Turing machines), which we achieve. For example, in the setting of delegation, a data owner wants to delegate her computation to a cloud, but the mere effort of encrypting the data is greater than computing the circuit directly, so the owner is better off doing the computation herself.

We remark that functional encryption (FE) arises from, and generalizes, a beautiful sequence of papers on attribute-based encryption (including [SW05, GPSW06, BSW07, GJPS08, LOS⁺10, Wat11, Wat12, LW12]), and more generally predicate encryption (including [BW07, KSW08, OT09]). We denote by attribute-based encryption (ABE) an encryption scheme where each ciphertext c of an underlying plaintext message m is tagged with a public attribute x . Each secret key sk_f is associated with a predicate f . Given a key sk_f and a ciphertext $c = \text{Enc}(x, m)$, the message m can be recovered if and only if $f(x)$ is true. Whether the message gets recovered or not, the attribute x is always public; in other words, the input to the computation of f , x , leaks with attribute-based encryption, whereas with functional encryption, nothing leaks about x other than $f(x)$. Therefore, attribute-based encryption offers qualitatively weaker security than functional encryption. Attribute-based encryption schemes were also called public-index predicate encryption schemes in the literature [BSW]. Boneh and Waters [BW07] introduced the idea of not leaking the attribute as in functional encryption (also called private-index functional encryption).

Very recently, the landscape of attribute-based encryption has significantly improved with the works of Gorbunov, Vaikuntanathan and Wee [GVW13], and Sahai and Waters [SW12], who construct attribute-based encryption schemes for general functions, and are a building block for our results.

1.1 Our Results

Our main result is the construction of a *succinct* single-key functional encryption scheme for *general functions*. We demonstrate the power of this result by showing that it can be used to address the long-standing open problem in cryptography of reusing garbled circuits, as well as making progress on other open problems.

We can state our main result as a reduction from *any* attribute-based encryption and *any* fully homomorphic encryption scheme. In particular, we show how to construct a (single-key and succinct) functional encryption scheme for any class of functions \mathcal{F} by using a homomorphic encryption scheme which can do homomorphic evaluations for any function in \mathcal{F} and an attribute-based encryption scheme for a

²These inner-product schemes allow an arbitrary number of keys.

³A universal circuit \mathcal{F} is a circuit that takes as input a description of a circuit f and an input string x , runs f on x and outputs $f(x)$.

⁴Namely, parameter q (the maximum number of keys allowed) is fixed during setup, and the ciphertexts size grows linearly with q .

“slightly larger” class of functions \mathcal{F}' ; \mathcal{F}' is the class of functions such that for any function $f \in \mathcal{F}$, the class \mathcal{F}' contains the function computing the i -th bit of the FHE evaluation of f .

Theorem 1.1 (Informal). *There is a single-key functional encryption scheme with succinct ciphertexts (independent of circuit size) for the class of functions \mathcal{F} assuming the existence of*

- *a fully homomorphic encryption scheme for the class of functions \mathcal{F} , and*
- *a (single-key) attribute-based encryption scheme for a class of predicates \mathcal{F}' (as above).*

The literature has considered two types of security for ABE and FE: selective and full security (see Sec. 2.6). We show that if the underlying ABE scheme is selectively or fully secure, our resulting FE scheme is selectively or fully secure, respectively.

Two very recent results achieve attribute-based encryption for general functions. Gorbunov, Vaikuntanathan and Wee [GVW13] achieve ABE for general circuits of bounded depth based on the subexponential Learning With Errors (LWE) intractability assumption. Sahai and Waters [SW12] achieve ABE for general circuits under the less standard k -Multilinear Decisional Diffie-Hellman (see [SW12] for more details); however, when instantiated with the only construction of multilinear maps currently known [GGH12], they also achieve ABE for general circuits of bounded depth. Our scheme can be instantiated with any of these schemes because our result is a reduction.

When coupling our theorem with the ABE result of [GVW13] and the FHE scheme of [BV11a, BGV12], we obtain:

Corollary 1.2 (Informal). *Under the subexponential LWE assumption, for any depth d , there is a single-key functional encryption scheme for general functions computable by circuits of depth d . The scheme has succinct ciphertexts: their size is polynomial in the depth d (and does not depend on the circuit size).*

This corollary holds for both selective and full security definitions, since [GVW13] constructs both selectively secure and fully secure ABE schemes. However, the parameters of the LWE assumption are different in the two cases (Sec. 2.3).

Another corollary of our theorem is that, given a *universal* ABE scheme (the scheme is for all classes of circuits, independent of depth) and any fully homomorphic encryption scheme, there is a universal functional encryption scheme whose ciphertext size does not depend on the circuit’s size or even the circuit’s depth.

As mentioned, extending our scheme to be secure against an adversary who receives q keys is straightforward. The basic idea is simply to repeat the scheme q times in parallel. This strategy results in the ciphertext size growing linearly with q , which is unavoidable for the simulation-based security definition we consider, because of the discussed impossibility result [AGVW12]. Stated in these terms, our scheme is also a q -collusion-resistant functional encryption scheme like [GVW12], but our scheme’s ciphertexts are succinct, whereas [GVW12]’s are proportional to the circuit size.

From now on, we restrict our attention to the single-key case, which is the essence of the new scheme. In the body of the paper we often omit the single-key or succinct adjectives and whenever we refer to a functional encryption scheme, we mean a succinct single-key functional encryption scheme.

We next show how to use our main theorem to make significant progress on some of the most intriguing open questions in cryptography today: the *reusability* of garbled circuits, a new paradigm for *general function obfuscation*, as well as applications to fully homomorphic encryption with evaluation running in *input-specific time* rather than in worst-case time, and to publicly verifiable delegation. Succinctness plays a central role in these applications and they would not be possible without it.

1.1.1 Main Application: Reusable Garbled Circuits

A circuit garbling scheme, which has been one of the most useful primitives in modern cryptography, is a construction originally suggested by Yao in the 80s in the context of secure two-party computation [Yao82]. This construction relies on the existence of a one-way function to encode an arbitrary circuit C (“garbling” the circuit) and then encode any input x to the circuit (where the size of the encoding is short, namely, it does not grow with the size of the circuit C); a party given the garbling of C and the encoding of x can run the garbled circuit on the encoded x and obtain $C(x)$. The most basic properties of garbled circuits are circuit and input privacy: an adversary learns nothing about the circuit C or the input x other than the result $C(x)$.

Over the years, garbled circuits and variants thereof have found many applications: two party secure protocols [Yao86], multi-party secure protocols [GMW87], one-time programs [GKR08], KDM-security [BHH10], verifiable computation [GGP10], homomorphic computations [GHV10] and others. However, a basic limitation of the original construction remains: it offers only *one-time* usage. Specifically, providing an encoding of more than one input compromises the secrecy of the circuit. Thus, evaluating the circuit C on any new input requires an entirely new garbling of the circuit.

The problem of reusing garbled circuits has been open for 30 years. Using our newly constructed succinct functional encryption scheme we are now able to build *reusable garbled circuits* that achieve *circuit and input privacy*: a garbled circuit for any computation of depth d (where the parameters of the scheme depend on d), which can be run on *any polynomial number* of inputs without compromising the privacy of the circuit or the input. More generally, we prove the following:

Theorem 1.3 (Informal). *There exists a polynomial p , such that for any depth function d , there is a reusable circuit garbling scheme for the class of all arithmetic circuits of depth d , assuming there is a single-key functional encryption scheme for all arithmetic circuits of depth $p(d)$.*⁵

Corollary 1.4 (Informal). *Under the subexponential LWE assumption, for any depth function d , there exists a reusable circuit garbling scheme with circuit and input privacy for all arithmetic circuits of depth d .*

Reusability of garbled circuits (for depth-bounded computations) implies a multitude of applications as evidenced by the research on garbled circuits over the last 30 years. We note that for many of these applications, depth-bounded computation suffices. We also note that some applications do not require circuit privacy. In that situation, our succinct single-key functional encryption scheme already provides reusable garbled circuits with input-privacy and, moreover, the encoding of the input is a public-key algorithm.

We remark that [GVW13] gives a restricted form of reusable circuit garbling: it provides authenticity of the circuit output, but does not provide input privacy or circuit privacy, as we do here. Informally, authenticity means that an adversary cannot obtain a different yet legitimate result from a garbled circuit. We note that most of the original garbling circuit applications (e.g., two party secure protocols [Yao86], multi-party secure protocols [GMW87]) rely on the privacy of the input or of the circuit.

One of the more intriguing applications of reusable garbled circuits pertains to a new model for program obfuscation, token-based obfuscation, which we discuss next.

1.1.2 Token-Based Obfuscation: a New Way to Circumvent Obfuscation Impossibility Results

Program obfuscation is the process of taking a program as input, and producing a functionally equivalent but different program, so that the new program reveals no information to a computationally bounded adversary

⁵For this application we need to assume that the underlying functional encryption scheme is fully secure (as opposed to only selectively secure).

about the original program, beyond what “black box access” to the program reveals. Whereas ad-hoc program obfuscators are built routinely, and are used in practice as the main software-based technique to fight reverse engineering of programs, in 2000 Barak et al. [BGI⁺01], followed by Goldwasser and Kalai [GK05], proved that program obfuscation for general functions is impossible using software alone, with respect to several strong but natural definitions of obfuscation.

The results of [BGI⁺01, GK05] mean that there exist functions which cannot be obfuscated. Still, the need to obfuscate or “garble” programs remains. A long array of works attempts to circumvent the impossibility results in various ways, including adding secure hardware components [GKR08, GIS⁺10, BCG⁺11], relaxing the definition of security [GR07], or considering only specific functions [Wee05, CKVW10].

The problem of obfuscation seems intimately related to the “garbled circuit” problem where given a garbling of a circuit C and an encoding for an input x , one can learn the result of $C(x)$ but nothing else. One cannot help but wonder whether the new reusable garbling scheme would *immediately* imply a solution for the obfuscation problem (which we know is impossible). Consider an example illustrating this intuition: a vendor obfuscates her program (circuit) by garbling it and then gives the garbled circuit to a customer. In order to run the program on (multiple) inputs x_i , the customer simply encodes the inputs according to the garbling scheme and thus is able to compute $C(x_i)$. Unfortunately, although close, this scenario does not work with reusable garbled circuits. The key observation is that encoding x requires knowledge of a secret key! Thus, an adversary cannot produce encoded inputs on its own, and needs to obtain “tokens” in the form of encrypted inputs from the data owner.

Instead, we propose a new *token-based* model for obfuscation. The idea is for a vendor to obfuscate an arbitrary program as well as provide tokens representing rights to run this program on specific inputs. For example, consider that some researchers want to obtain statistics out of an obfuscated database containing sensitive information (the obfuscated program is the program running queries with the secret database hardcoded in it). Whenever the researchers want to input a query x to this program, they need to obtain a token for x from the program owner. To produce each token, the program owner does little work. The researchers perform the bulk of the computation by themselves using the token and obtain the computation result without further interaction with the owner.

Claim 1.5. *Assuming a reusable garbling scheme for a class of circuits, there is a token-based obfuscation scheme for the same class of circuits.*

Corollary 1.6 (Informal). *Under the subexponential LWE assumption, for any depth function d , there exists a token-based obfuscation scheme for all arithmetic circuits of depth d .*

It is worthwhile to compare the token-based obfuscation model with previous work addressing obfuscation using trusted-hardware components such as [GIS⁺10, BCG⁺11]. In these schemes, after a user finishes executing the obfuscated program on an input, the user needs to interact with the trusted hardware to obtain the decryption of the result; in comparison, in our scheme, the user needs to obtain only a token before the computation begins, and can then run the computation and obtain the decrypted result by herself.

1.1.3 Computing on Encrypted Data in Input-Specific Time

All current FHE constructions work according to the following template. For a fixed input size, a program is transformed into an arithmetic circuit; homomorphic evaluation happens gate by gate on this circuit. The size of the circuit reflects the *worst-case* running time of the program: for example, every loop is unfolded into the maximum number of steps corresponding to the worst-case input, and each function is called the

maximum number of times possible. Such a circuit can be potentially very large, despite the fact that there could be many inputs on which the execution is short.

A fascinating open question has been whether it is possible to perform FHE following a Turing-machine-like template: *the computation time is input-specific* and can terminate earlier depending on the input at hand. Of course, to compute in input-specific time, the running time must unavoidably leak to the evaluator, but such leakage is acceptable in certain applications and the efficiency gains can be significant; therefore, such a scheme provides weaker security than fully homomorphic encryption (namely, nothing other than the running time leaks about the input), at the increase of efficiency.

Using our functional encryption scheme, we show how to achieve this goal. The idea is to use the scheme to test when an encrypted circuit computation has terminated, so the computation can stop earlier on certain inputs. We overview our technique in Sec. 1.2.

Because the ciphertexts in our functional encryption scheme grow with the depth of the circuits, such a scheme is useful only for Turing machines that can be expressed as circuits of depth at most $d(n)$ for inputs of size n . We refer to such Turing machines as *d-depth-bounded* and define them in Sec. 6.

Theorem 1.7. *There is a scheme for evaluating Turing machines on encrypted inputs in input-specific time for any class of d-depth-bounded Turing machines, assuming the existence of a succinct single-key functional encryption scheme for circuits of depth d ,⁶ and a fully homomorphic encryption scheme for circuits of depth d .*

Corollary 1.8. *Under the subexponential LWE assumption, for any depth d , there is a scheme for evaluating Turing machines on encrypted data in input-specific time for any class of d-depth-bounded Turing machines.*

1.1.4 Publicly Verifiable Delegation with Secrecy

Recently, Parno, Raykova and Vaikuntanathan [PRV12] showed how to construct a 2-message delegation scheme that is *publicly verifiable*, in the preprocessing model, from any attribute-based encryption scheme. This reduction can be combined with [GVW13]’s ABE scheme to achieve such a delegation scheme.

However, this scheme does not provide *secrecy* of the inputs: the prover can learn the inputs. By replacing the ABE scheme in the construction of [PRV12] with our new functional encryption scheme, we add secrecy to the scheme; namely, we obtain a delegation scheme which is both *publicly verifiable* as in [PRV12] (anyone can verify that a transcript is accepting using only public information) and *secret* (the prover does not learn anything about the input of the function being delegated).⁷ More specifically, we construct a 2-message delegation scheme in the preprocessing model that is based on the subexponential LWE assumption, and is for general depth-bounded circuits, where the verifier works in time that depends on the *depth* of the circuit being delegated, but is independent of the size of the circuit, and the prover works in time dependent on the *size* of the circuit.

1.2 Technique Outline

Our functional encryption scheme. We first describe the ideas behind our main technical result: a reduction from attribute-based encryption (ABE) and fully homomorphic encryption (FHE) to functional encryption (FE).

⁶As in previous applications, we need to assume that the underlying functional encryption scheme is fully secure (as opposed to only selectively secure).

⁷We note that secrecy can be easily obtained by using an FHE scheme, however, this destroys public-verifiability.

Compute on encrypted data with FHE. A natural starting point is FHE because it enables computation on encrypted data, which is needed with functional encryption. Using FHE, the FE encryption of an input x consists of an FHE encryption of x , denoted \hat{x} , while the secret key for a function f is simply f itself. The semantic security of FHE provides the desired security (and more) because nothing leaks about x ; however, using FHE evaluation, the evaluator obtains an encrypted computation result, $\widehat{f(x)}$, instead of the decrypted value $f(x)$. Giving the evaluator the FHE decryption key is not an option because the evaluator can use it to decrypt x as well.

Attempt to decrypt using a Yao garbled circuit. We would like the evaluator to decrypt the FHE ciphertext $\widehat{f(x)}$, but not be able to decrypt anything else. An idea is for the owner to give the evaluator a Yao garbled circuit for the FHE decryption function FHE.Dec with the FHE secret key hsk hardcoded in it, namely a garbled circuit for $\text{FHE.Dec}_{\text{hsk}}$. When the owner garbles $\text{FHE.Dec}_{\text{hsk}}$, the owner also obtains a set of garbled circuit labels $\{L_0^i, L_1^i\}_i$. The evaluator must only receive the input labels corresponding to $\widehat{f(x)}$: namely, the labels $\{L_{b_i}^i\}_i$ where b_i is the i -th bit of $\widehat{f(x)}$. But this is not possible because the owner does not know a priori $\widehat{f(x)}$ which is determined only after the FHE evaluation; furthermore, after providing more than one set of labels (which happens when encrypting another input x'), the security of the garbled circuit (and hence of the FHE secret key) is compromised. One idea is to have the owner and the evaluator interact, but the syntax of functional encryption does not allow interaction. Therefore, the evaluator needs to determine the set of labels corresponding to $\widehat{f(x)}$ by herself, and *should not obtain any other labels*.

Constraining decryption using ABE. It turns out that what we need here is very close to what ABE provides. Consider the following variant of ABE (called ABE_2) that can be constructed easily from a standard ABE scheme. One encrypts a value y together with two messages m_0, m_1 and obtains a ciphertext $c \leftarrow \text{ABE}_2.\text{Enc}(y, m_0, m_1)$. Then, one generates a key for a predicate g : $\text{sk}_g \leftarrow \text{ABE}_2.\text{KeyGen}(g)$. The decryption algorithm on input c and sk_g outputs m_0 if $g(y) = 0$ or outputs m_1 if $g(y) = 1$.

Now consider using ABE_2 multiple times, once for every $i \in \{1, \dots, \text{size of } \widehat{f(x)}\}$. For the i -th invocation of $\text{ABE}_2.\text{Enc}$, let m_0, m_1 be the garbled labels L_0^i, L_1^i , and let y be \hat{x} : $\text{ABE}_2.\text{Enc}(\hat{x}, L_0^i, L_1^i)$. Next, for the i -th invocation of $\text{ABE}_2.\text{KeyGen}$, let g be FHE.Eval_f^i (the predicate returning the i -th bit of the evaluation of f on an input ciphertext): $\text{ABE}_2.\text{KeyGen}(\text{FHE.Eval}_f^i)$. Then, the evaluator can use $\text{ABE}_2.\text{Dec}$ to obtain the needed label: $L_{b_i}^i$ where b_i is the i -th bit of $\widehat{f(x)}$. Armed with these labels and the garbled circuit, the evaluator decrypts $\widehat{f(x)}$.

The security of the ABE scheme ensures the evaluator cannot decrypt any other labels, so the evaluator cannot learn more than $\widehat{f(x)}$. Finally, note that the one-time aspect of garbled circuits does not restrict the number of encryptions with our FE scheme because the encryption algorithm generates a new garbled circuit every time; since the garbled circuit is for the FHE decryption algorithm (which is a fixed algorithm), the size of the ciphertexts remains independent of the size of f .

We now explain how to use this result to obtain the aforementioned applications.

From FE to reusable garbled circuits. The goal of garbled circuits is to hide the input and the circuit C . Our succinct single-key FE already provides a reusable garbling scheme with input privacy (the single key corresponds to the circuit to garble). To obtain circuit privacy, the insight is to leverage the secrecy of the inputs to hide the circuit. The first idea that comes to mind is to generate a key for the universal circuit instead of C , and include C in the ciphertext when encrypting an input. However, this approach will yield large ciphertexts, as large as the circuit size.

Instead, the insight is to garble C by using a semantically secure encryption scheme E.Enc together with our FE scheme: the garbling of C will be an FE secret key for a circuit U that contains $\text{E.Enc}_{\text{sk}}(C)$; on

input (sk, x) , U uses sk to decrypt C and then runs C on the input x . The token for an input x will be an FE encryption of (sk, x) . Now, even if the FE scheme does not hide $E.Enc_{sk}(C)$, the security of the encryption scheme E hides C .

Computing on encrypted data in input-specific time. We now summarize our approach to evaluating a Turing machine (TM) M homomorphically over encrypted data without running in worst-case time on all inputs. Sec. 6 presents the scheme formally.

Our idea is to use our functional encryption scheme to enable the evaluator to determine at various intermediary steps in the evaluation whether the computation finished or not. For each intermediary step, the client provides a secret key for a function that returns a bit indicating whether the computation finished or not. However, if the client provides a key for every computation step, then the amount of keys corresponds to the worst-case running time. Thus, instead, we choose intermediary points spaced at exponentially increasing intervals. In this way, the client generates only a logarithmic number of keys, namely for functions indicating if the computation finishes in $1, 2, 4, \dots, 2^i, \dots, 2^{\lceil \log t_{\max} \rceil}$ steps, where t_{\max} is the worst-case running time of M on all inputs of a certain size.

Because of the single-key aspect of our FE scheme, the client cannot provide keys for an arbitrary number of TMs to the evaluator. However, this does not mean that the evaluator can run only an a priori fixed number of TMs on the encrypted data. The reason is that the client can provide keys for the universal TMs $U_0, \dots, U_{\lceil \log t_{\max} \rceil}$, where TM U_i is the TM that on input a TM M and a value x , runs M on x for 2^i steps and outputs whether M finished.

Therefore, in an offline preprocessing phase, the client provides $1 + \lceil \log t_{\max} \rceil$ keys where the i -th key is for a circuit corresponding to U_i , each key being generated with a different master secret key. The work of the client in this phase is at least t_{\max} which is costly, but this work happens only once and is amortized over all subsequent inputs in the online phase.

In an online phase, the client receives an input x and wants the evaluator to compute $M(x)$ for her. The client provides FE encryptions of (M, x) to the evaluator together with an FHE ciphertext (\hat{M}, \hat{x}) for (M, x) to be used for a separate FHE evaluation. The evaluator tries each key sk_{U_i} from the preprocessing phase and learns the smallest i for which the computation of M on x stops in 2^i steps. The evaluator then computes a universal circuit of size $\tilde{O}(2^i)$ and evaluates it homomorphically over (\hat{M}, \hat{x}) , obtaining the FHE encryption of $M(x)$. Thus, we can see that the evaluator runs in time polynomial in the runtime of M on x .

Publicly Verifiable Delegation with Secrecy. Delegation schemes aim to enable a weak verifier to delegate computation of a function f on an input x to a prover who can then prove to the verifier that he computed the function correctly. We now show that our single-key functional encryption scheme provides an improvement to publicly verifiable delegation by adding secrecy. We present this improvement only informally, because we prefer to focus on the other applications.

We now briefly recall the scheme of [PRV12] and then discuss how to modify it; we refer the reader to Section 2.6 for formal definitions of ABE and FE. There are two phases in the delegation scheme: the preprocessing phase when the verifier prepares the computation f , and an online phase repeating many times, in which the verifier gives x to the prover who computes $f(x)$ and proves the computation was correct.

In the preprocessing phase, the verifier generates two pairs of master secret and public keys (msk_1, mpk_1) and (msk_2, mpk_2) for the underlying attribute-based encryption scheme. If f is the function to delegate, the verifier uses msk_1 to generate a key for f denoted sk_f , and msk_2 to generate a key for the negation of f , $\bar{f}(x) := 1 - f(x)$, denoted $sk_{\bar{f}}$. The verifier then sends both (mpk_1, mpk_2) and $(sk_f, sk_{\bar{f}})$ to the prover. Generating sk_f and $sk_{\bar{f}}$ takes time that is proportional to the size of the circuit computing f , and thus is a costly operation. However, this is done only once in the preprocessing phase.

Whenever the verifier wants the prover to compute f on an input x , he chooses two random messages

m_1, m_2 and sends the prover the encryptions of (x, m_*) under the two keys: $(\text{Enc}(\text{mpk}_1, x, m_1))$ and $(\text{Enc}(\text{mpk}_2, x, m_2))$. The properties of the attribute-based encryption scheme guarantees that, if $f(x) = 1$, the prover obtains m_1 using sk_f and \perp using $\text{sk}_{\bar{f}}$ so no information about m_0 , and vice versa if $f(x) = 0$. Therefore, the fact that the prover provides m_1 to the verifier is a proof that $f(x)$ was 1.

Importantly, this delegation scheme can be made to have the desired property of being *publicly verifiable*, meaning that the verifier can produce a “verification key” with which anyone can check the prover’s work. This is done by having the verifier also send two point function obfuscations, one of the point m_1 and the other of the point m_2 .

This reduction from ABE to publicly verifiable delegation can be combined with the recent result of [GVW13] providing ABE schemes for any depth circuit: the result is a publicly verifiable 2-message delegation scheme in the preprocessing model for any depth d circuit with verifier’s work being proportional to the depth d and the prover’s work proportional to the circuit size.

Note however, that this scheme is not secret because ABE does not hide the input x from the prover. It is well known that x can be made secret by encrypting everything using a fully homomorphic encryption scheme. However, this comes at the cost of losing the public verifiability property. Our idea is to replace the ABE scheme with our functional encryption scheme in the protocol above; now the ciphertexts $\text{Enc}(\text{mpk}_1, x, m_1)$ and $\text{Enc}(\text{mpk}_2, x, m_2)$ hide x and the scheme provides secrecy because the prover learns nothing about x other than $f(x)$. The public verifiability of the scheme remains the same.

We remark that we could provide a stronger version of secrecy by also hiding the result $f(x)$ from the prover; such stronger secrecy is non-standard for delegation, so we do not delve on it. (The idea is for the client to concatenate a random bit to each input x and have the function f output the opposite result when the bit is set. In this way, the prover does not learn anything from seeing which ciphertext decrypts to non- \perp .)

2 Preliminaries

2.1 Notation

Let κ denote the security parameter throughout this paper. For a distribution \mathcal{D} , we say $x \leftarrow \mathcal{D}$ when x is sampled from the distribution \mathcal{D} . If S is a finite set, by $x \leftarrow S$ we mean x is sampled from the uniform distribution over the set S . We use $p(\cdot)$ to denote that p is a function that takes one input. Similarly, $p(\cdot, \cdot)$ denotes a function p that takes two inputs.

We say that a function f is negligible in an input parameter κ , if for all $d > 0$, there exists K such that for all $\kappa > K$, $f(\kappa) < \kappa^{-d}$. For brevity, we write: for all sufficiently large κ , $f(\kappa) = \text{negl}(\kappa)$. We say that a function f is polynomial in an input parameter κ , if there exists a polynomial p such that for all κ , $f(\kappa) \leq p(\kappa)$. We write $f(\kappa) = \text{poly}(\kappa)$. A similar definition holds for $\text{polylog}(\kappa)$.

Let $[n]$ denote the set $\{1, \dots, n\}$ for $n \in \mathbb{N}^*$. When saying that a Turing machine A is p.p.t. we mean that A is a non-uniform probabilistic polynomial-time machine.

In this paper, we only work with arithmetic circuits over $\text{GF}(2)$. These circuits have two types of gates: $+$ mod 2 and \times mod 2. Unless the context specifies otherwise, we consider circuits with one bit of output (also called boolean).

Two ensembles, $X = \{X_\kappa\}_{\kappa \in \mathbb{N}}$ and $Y = \{Y_\kappa\}_{\kappa \in \mathbb{N}}$, are said to be *computationally indistinguishable* (and denoted $\{X_\kappa\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx} \{Y_\kappa\}_{\kappa \in \mathbb{N}}$) if for every probabilistic polynomial-time algorithm D ,

$$|\Pr[D(X_\kappa, 1^\kappa) = 1] - \Pr[D(Y_\kappa, 1^\kappa) = 1]| = \text{negl}(\kappa).$$

In our security definitions, we will define probabilistic experiments and denote by random variables their

outputs. For example, $\text{Exp}_{E,A}^{\text{real}}(1^\kappa)$ denotes the random variable representing the output of the real experiment for scheme E with adversary A on security parameter κ . Moreover, $\{\text{Exp}_{E,A}^{\text{real}}(1^\kappa)\}_{\kappa \in \mathbb{N}}$ denotes the ensemble of such random variables indexed by $\kappa \in \mathbb{N}$.

2.2 Background on Learning With Errors (LWE)

The security of our results will be based on the Learning with Errors (LWE) assumption, first introduced by Regev [Reg05]. Regev showed that solving the LWE problem *on average* is (quantumly) as hard as solving the approximate version of several standard lattice problems, such as *gapSVP in the worst case*. Peikert [Pei09] later removed the quantum assumption from a variant of this reduction. Given this connection, we state all our results under worst-case lattice assumptions, and in particular, under (a variant of) the gapSVP assumption. We refer the reader to [Reg05, Pei09] for details about the worst-case/average-case connection.

The best known algorithms to solve these lattice problems with an approximation factor 2^{ℓ^ϵ} in ℓ -dimensional lattices run in time $2^{\tilde{O}(\ell^{1-\epsilon})}$ [AKS01, MV10] for any constant $0 < \epsilon < 1$. Specifically, given the current state-of-the-art on lattice algorithms, it is quite plausible that achieving approximation factors 2^{ℓ^ϵ} for these lattice problems is hard for polynomial time algorithms.

Appendix A provides more detailed background information on LWE.

2.3 Fully Homomorphic Encryption (FHE)

The notion of fully homomorphic encryption was first proposed by Rivest, Adleman and Dertouzos [RAD78] in 1978. The first fully homomorphic encryption scheme was proposed in a breakthrough work by Gentry in 2009 [Gen09]. A history and recent developments on fully homomorphic encryption is surveyed in [Vai11]. We recall the definitions and semantic security of fully homomorphic encryption; the definitions below are based on [Vai11] with some adaptations.

Definition 2.1. *A homomorphic (public-key) encryption scheme FHE is a quadruple of polynomial time algorithms (FHE.KeyGen, FHE.Enc, FHE.Dec, FHE.Eval) as follows:*

- $\text{FHE.KeyGen}(1^\kappa)$ is a probabilistic algorithm that takes as input the security parameter 1^κ and outputs a public key pk and a secret key sk .
- $\text{FHE.Enc}(\text{pk}, x \in \{0, 1\})$ is a probabilistic algorithm that takes as input the public key pk and an input bit x and outputs a ciphertext ψ .
- $\text{FHE.Dec}(\text{sk}, \psi)$ is a deterministic algorithm that takes as input the secret key sk and a ciphertext ψ and outputs a message $x^* \in \{0, 1\}$.
- $\text{FHE.Eval}(\text{pk}, C, \psi_1, \psi_2, \dots, \psi_n)$ is a deterministic algorithm that takes as input the public key pk , some circuit C that takes n bits as input and outputs one bit, as well as n ciphertexts ψ_1, \dots, ψ_n . It outputs a ciphertext ψ_C .

Compactness: *For all security parameters κ , there exists a polynomial $p(\cdot)$ such that for all input sizes n , for all $x_1 \dots x_n$, for all C , the output length of FHE.Eval is at most $p(n)$ bits long.*

Definition 2.2 (C-homomorphism). *Let $\mathcal{C} = \{C_n\}_{n \in \mathbb{N}}$ be a class of boolean circuits, where C_n is a set of boolean circuits taking n bits as input. A scheme FHE is C-homomorphic if for every polynomial $n(\cdot)$, for*

every sufficiently large security parameter κ , for every circuit $C \in \mathcal{C}_n$, and for every input bit sequence x_1, \dots, x_n , where $n = n(\kappa)$,

$$\begin{aligned} & \Pr[(pk, sk) \leftarrow \text{FHE.KeyGen}(1^\kappa); \\ & \quad \psi_i \leftarrow \text{FHE.Enc}(pk, x_i) \text{ for } i = 1 \dots n; \\ & \quad \psi \leftarrow \text{FHE.Eval}(pk, C, \psi_1, \dots, \psi_n) : \\ & \quad \text{FHE.Dec}(sk, \psi) \neq C(x_1, \dots, x_n)] = \text{negl}(\kappa). \end{aligned}$$

where the probability is over the coin tosses of FHE.KeyGen and FHE.Enc .

Definition 2.3 (Fully homomorphic encryption). *A scheme FHE is fully homomorphic if it is homomorphic for the class of all arithmetic circuits over $\text{GF}(2)$.*

Definition 2.4 (Leveled fully homomorphic encryption). *A leveled fully homomorphic encryption scheme is a homomorphic scheme where FHE.KeyGen receives an additional input 1^d and the resulting scheme is homomorphic for all depth- d arithmetic circuits over $\text{GF}(2)$.*

Definition 2.5 (IND-CPA security). *A scheme FHE is IND-CPA secure if for any p.p.t. adversary \mathcal{A} ,*

$$\begin{aligned} & \left| \Pr[(pk, sk) \leftarrow \text{FHE.KeyGen}(1^\kappa) : \mathcal{A}(pk, \text{FHE.Enc}(pk, 0)) = 1] - \right. \\ & \left. \Pr[(pk, sk) \leftarrow \text{FHE.KeyGen}(1^\kappa) : \mathcal{A}(pk, \text{FHE.Enc}(pk, 1)) = 1] \right| = \text{negl}(\kappa). \end{aligned}$$

We now state the result of Brakerski, Gentry and Vaikuntanathan [BGV12] that shows a leveled fully homomorphic encryption scheme based on the LWE assumption:

Theorem 2.1 ([BV11a, BGV12]). *Assume that there is a constant $0 < \epsilon < 1$ such that for every sufficiently large ℓ , the approximate shortest vector problem gapSVP in ℓ dimensions is hard to approximate to within a $2^{O(\ell^\epsilon)}$ factor in the worst case. Then, for every n and every polynomial $d = d(n)$, there is an IND-CPA secure d -leveled fully homomorphic encryption scheme where encrypting n bits produces ciphertexts of length $\text{poly}(n, \kappa, d^{1/\epsilon})$, the size of the circuit for homomorphic evaluation of a function f is $\text{size}(C_f) \cdot \text{poly}(n, \kappa, d^{1/\epsilon})$ and its depth is $\text{depth}(C_f) \cdot \text{poly}(\log n, \log d)$.*

All known fully homomorphic encryption schemes (as opposed to merely leveled schemes) require an additional assumption related to circular security of the associated encryption schemes. However, we do not need to make such an assumption in this work because we only use a leveled homomorphic encryption scheme in our constructions.

2.4 Background on Garbled Circuits

We will now define garbled circuits. Initially, garbled circuits were presented by Yao [Yao82] in the context of secure two-party computation and later, they were then proven secure by Lindell and Pinkas [LP09]. Very recently, the notion has been formalized by Bellare et al. [BHR12]. For simplicity, we present more concise definitions of garbled circuits than in [BHR12].

Definition 2.6 (Garbling scheme). *A garbling scheme for a family of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$ with \mathcal{C}_n a set of boolean circuits taking as input n bits, is a tuple of p.p.t. algorithms $\text{Gb} = (\text{Gb.Garble}, \text{Gb.Enc}, \text{Gb.Eval})$ such that*

- $\text{Gb.Garble}(1^\kappa, C)$ takes as input the security parameter κ and a circuit $C \in \mathcal{C}_n$ for some n , and outputs the garbled circuit Γ and a secret key sk .
- $\text{Gb.Enc}(\text{sk}, x)$ takes as input $x \in \{0, 1\}^*$ and outputs an encoding c .
- $\text{Gb.Eval}(\Gamma, c)$ takes as input a garbled circuit Γ , an encoding c and outputs a value y which should be $C(x)$.

Correctness. For any polynomial $n(\cdot)$, for all sufficiently large security parameters κ , for $n = n(\kappa)$, for all circuits $C \in \mathcal{C}_n$ and all $x \in \{0, 1\}^n$,

$$\Pr[(\Gamma, \text{sk}) \leftarrow \text{Gb.Garble}(1^\kappa, C); c \leftarrow \text{Gb.Enc}(\text{sk}, x); y \leftarrow \text{Gb.Eval}(\Gamma, c) : C(x) = y] = 1 - \text{negl}(\kappa).$$

Efficiency. There exists a universal polynomial $p = p(\kappa, n)$ (p is the same for all classes of circuits \mathcal{C}) such that for all input sizes n , security parameters κ , for all boolean circuits C of with n bits of input, for all $x \in \{0, 1\}^n$,

$$\Pr[(\Gamma, \text{sk}) \leftarrow \text{Gb.Garble}(1^\kappa, C) : |\text{sk}| \leq p(\kappa, n) \text{ and } \text{runtime}(\text{Gb.Enc}(\text{sk}, x)) \leq p(\kappa, n)] = 1.$$

Note that since Gb.Enc is a p.p.t. algorithm, it suffices to ensure that $|\text{sk}| \leq p(\kappa, n)$ and obtain that Gb.Enc 's runtime is also at most a polynomial. We prefer to keep the runtime of Gb.Enc in the definition as well for clarity.

Remark 2.2 (Remark on the efficiency property). *Intuitively, a garbling scheme is efficient if the time to encode is shorter than the time to run the circuit. This requirement can be formalized in a few ways. A first definition is as provided above in Def. 2.6. Another definition is to allow $|\text{sk}|$ and the runtime of Gb.Enc to also depend on the depth of the circuits in \mathcal{C} , but require that it does not depend on their size.*

Yao garbled circuits. The garbled circuits presented by Yao have a specific property of the encoding scheme that is useful in various secure function evaluation protocols and in our construction as well. The secret key is of the form $\text{sk} = \{L_i^0, L_i^1\}_{i=1}^n$ and the encoding of an input x of n bits is of the form $c = (L_1^{x_1}, \dots, L_n^{x_n})$, where x_i is the i -th bit of x .

Two security guarantees are of interest: input privacy (the input to the garbled circuit does not leak to the adversary), and circuit privacy (the circuit does not leak to the adversary). All these properties hold only for one-time evaluation of the circuit: the adversary can receive at most one encoding of an input to use with a garbled circuit; obtaining more than one encoding breaks these security guarantees.

Bellare et al. [BHR12] also present a third property which they call authenticity; informally, this requires that an adversary should not be able to come up with a different result of the garbled circuit that could be “de-garbled” into a valid value. We do not present this property here because it is straightforward to show that a garbling scheme with input and circuit privacy as we define them below implies a different garbling scheme with the authenticity property and we would need to provide a slightly more complicated syntax for the definition of garbled circuits (with an additional “de-garbling” algorithm).

We now present the one-time security of garbling circuits. The security definition for reusable garbled will be presented later, in Sec. 4.

Definition 2.7 (Input and circuit privacy). *A garbling scheme Gb for a family of circuits $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$ is input and circuit private if there exists a p.p.t. simulator $\text{Sim}_{\text{Garble}}$, such that for every p.p.t. adversaries A and D , for all sufficiently large security parameters κ ,*

$$\begin{aligned} & \Pr[(x, C, \alpha) \leftarrow A(1^\kappa); (\Gamma, \text{sk}) \leftarrow \text{Gb.Garble}(1^\kappa, C); c \leftarrow \text{Gb.Enc}(\text{sk}, x) : D(\alpha, x, C, \Gamma, c) = 1] - \\ & \Pr[(x, C, \alpha) \leftarrow A(1^\kappa); (\tilde{\Gamma}, \tilde{c}) \leftarrow \text{Sim}_{\text{Garble}}(1^\kappa, C(x), 1^{|C|}, 1^{|x|}) : D(\alpha, x, C, \tilde{\Gamma}, \tilde{c}) = 1] = \text{negl}(\kappa), \end{aligned}$$

where we consider only A such that for some n , $x \in \{0, 1\}^n$ and $C \in \mathcal{C}_n$.

Intuitively, this definition says that, for any circuit or input chosen adversarially, one can simulate in polynomial time the garbled circuit and the encoding solely based on the computation result (and relevant sizes). The variable α represents any state that A may want to convey to D .

A few variants of Yao garbling schemes exist (for example, [BHR12]) that provide both input and circuit privacy under the basic one-way function assumption. Any such construction is suitable for our scheme.

Theorem 2.3 ([Yao82, LP09]). *Assuming one-way functions exist, there exists a Yao (one-time) garbling scheme that is input- and circuit-private for all circuits over $GF(2)$.*

2.5 Attribute-Based Encryption (ABE)

We now provide the definition of attribute-based encryption from the literature (e.g., [GPSW06, LOS⁺10, GVW13]).

Definition 2.8 (Attribute-Based Encryption). *An attribute-based encryption scheme (ABE) for a class of predicates $\mathcal{P} = \{P_n\}_{n \in \mathbb{N}}$ represented as boolean circuits with n input bits and one output bit and an associated message space \mathcal{M} is a tuple of algorithms (ABE.Setup, ABE.KeyGen, ABE.Enc, ABE.Dec) as follows:*

- $\text{ABE.Setup}(1^\kappa)$: Takes as input a security parameter 1^κ and outputs a public master key fmpk and a master secret key fmsk .
- $\text{ABE.KeyGen}(\text{fmsk}, P)$: Given a master secret key fmsk and a predicate $P \in \mathcal{P}_n$, for some n , outputs a key fsk_P corresponding to P .
- $\text{ABE.Enc}(\text{fmpk}, x, M)$: Takes as input the public key fmpk , an attribute $x \in \{0, 1\}^n$, for some n , and a message $M \in \mathcal{M}$ and outputs a ciphertext c .
- $\text{ABE.Dec}(\text{fsk}_P, c)$: Takes as input a secret key for a predicate and a ciphertext and outputs $M^* \in \mathcal{M}$.

Correctness. For any polynomial $n(\cdot)$, for every sufficiently large security parameter κ , if $n = n(\kappa)$, for all predicates $P \in \mathcal{P}_n$, attributes $x \in \{0, 1\}^n$, and messages $M \in \mathcal{M}$:

$$\Pr \left[\begin{array}{l} (\text{fmpk}, \text{fmsk}) \leftarrow \text{ABE.Setup}(1^\kappa); \\ \text{fsk}_P \leftarrow \text{ABE.KeyGen}(\text{fmsk}, P); \\ c \leftarrow \text{ABE.Enc}(\text{fmpk}, x, M); \\ \text{ABE.Dec}(\text{fsk}_P, c) = \begin{cases} M, & \text{if } P(x) = 1, \\ \perp, & \text{otherwise.} \end{cases} \end{array} \right] = 1 - \text{negl}(\kappa).$$

The space $\{0, 1\}^n$ is referred to as the *attribute space* (with an attribute size of n) and \mathcal{M} is referred to as the *message space*.

Intuitively, the security of ABE is that M is revealed only if $P(x) = 1$. Regarding the attribute x , ABE's security does not require any secrecy of the attribute, so x may leak no matter what is the value of $P(x)$. Many ABE schemes have been proven secure under indistinguishability-based definitions. Despite being

weaker than simulation-based definitions, such definitions suffice for the security of our construction, so we present them here. Two notions of security have been used in previous work: full and selective security. Full security allows the adversary to provide the challenge ciphertext after seeing the public key, whereas in selective security, the adversary must provide the challenge ciphertext before seeing the public key. We present both in the full security and selective security cases, because the ABE primitive we use [GVW13] achieves them with different parameters of the gapSVP assumption. We only provide the security definition for the case when the adversary can ask for *a single key* because this is all we need for our results.

Definition 2.9 (Attribute-based encryption security). *Let ABE be an attribute-based encryption scheme for a class of predicates $\mathcal{P} = \{\mathcal{P}_n\}_{n \in \mathbb{N}}$, and an associated message space \mathcal{M} , and let $A = (A_1, A_2, A_3)$ be a triple of p.p.t. adversaries. Consider the following experiment.*

$\text{Exp}_{\text{ABE}}(1^\kappa)$:

- 1: $(\text{fmpk}, \text{fmsk}) \leftarrow \text{ABE.Setup}(1^\kappa)$
 - 2: $(P, \text{state}_1) \leftarrow A_1(\text{fmpk})$
 - 3: $\text{fsk}_P \leftarrow \text{ABE.KeyGen}(\text{fmsk}, P)$
 - 4: $(M_0, M_1, x, \text{state}_2) \leftarrow A_2(\text{state}_1, \text{fsk}_P)$
 - 5: Choose a bit b at random and let $c \leftarrow \text{ABE.Enc}(\text{fmpk}, x, M_b)$.
 - 6: $b' \leftarrow A_3(\text{state}_2, c)$. If $|M_0| = |M_1|$, $P(x) = 0$, and $b = b'$, output 1, else output 0.
-

We say that the scheme is a single-key fully-secure attribute-based encryption if for all p.p.t. adversaries A , and for all sufficiently large κ :

$$\Pr[\text{Exp}_{\text{ABE}, A}(1^\kappa) = 1] \leq 1/2 + \text{negl}(\kappa).$$

We say that the scheme is single-key selectively secure if the same statement holds for a slightly modified game in which A provides x before receiving fmpk .

Attribute-based encryption schemes have been constructed for the class of Boolean formulas [GPSW06, LOS⁺10] and most recently for the class of all polynomial-size circuits: Gorbunov, Vaikuntanathan and Wee [GVW13] based on the subexponential Learning With Errors (LWE) intractability assumption, and Sahai and Waters [SW12] based on the k -Multilinear Decisional Diffie-Hellman (see [SW12] for more details). Our reduction can start from any of these schemes, but in this paper, we choose [GVW13] because it is based on LWE, which is a more standard assumption and is also the assumption for our other building block, FHE.

Before we state the results of Gorbunov, Vaikuntanathan and Wee [GVW13], we will set up some notation. Let d and p be two univariate polynomials. Define $\mathcal{C}_{n, d(n), p(n)}$ to be the class of all boolean circuits on n inputs of depth at most $d(n)$ and size at most $p(n)$. Let $\mathcal{C}_{n, d(n)} := \bigcup_{\text{polynomial } p} \mathcal{C}_{n, d(n), p(n)}$. An attribute-based encryption or functional encryption scheme that supports circuits in $\mathcal{C}_{n, d(n)}$ is called a d -leveled attribute-based encryption or functional encryption scheme, respectively. We also refer to an ABE or FE scheme as leveled, if it is d -leveled for some d . We are now ready to state the theorem of [GVW13].

Theorem 2.4 ([GVW13]). *Assume that there is a constant $0 < \epsilon < 1$ such that for every sufficiently large ℓ , the approximate shortest vector problem gapSVP in ℓ dimensions is hard to approximate by a polynomial algorithm to within a $2^{O(\ell^\epsilon)}$ factor in the worst case. Then, for every n and every polynomial $d = d(n)$, there is a selectively secure d -leveled attribute-based encryption scheme where encrypting n bits produces ciphertexts of length $\text{poly}(n, \kappa, d^{1/\epsilon})$.*

Furthermore, assuming that gapSVP in ℓ dimensions is hard to approximate to within a $2^{O(\ell^\epsilon)}$ factor in time $2^{O(\ell^\epsilon)}$, the scheme is fully secure with ciphertexts of length $\text{poly}(n, \kappa, d^{1/\epsilon^2})$.

In either case, the scheme is secure with polynomially many secret-key queries.

2.5.1 Two-Outcome Attribute-Based Encryption

We use an attribute-based encryption scheme with a slightly modified definition. The setup and key generation algorithms are the same as in previous schemes. The difference is in the encryption and decryption algorithms: instead of encrypting one message M in one ciphertext, we encrypt two messages M_0 and M_1 in the same ciphertext such that M_0 is revealed if the predicate evaluates to zero on the attribute, and M_1 is revealed if the predicate evaluates to one. Since there are two possible outcomes of the decryption algorithm, we call the modified scheme a *two-outcome attribute-based encryption* scheme. Such a variant of ABE has been used for other purposes by [PRV12].

Definition 2.10 (Two-Outcome Attribute-Based Encryption). *A two-outcome attribute-based encryption scheme (ABE_2) for a class of predicates $\mathcal{P} = \{P_n\}_{n \in \mathbb{N}}$ represented as boolean circuits with n input bits, and a message space \mathcal{M} is a tuple of algorithms ($\text{ABE}_2.\text{Setup}$, $\text{ABE}_2.\text{KeyGen}$, $\text{ABE}_2.\text{Enc}$, $\text{ABE}_2.\text{Dec}$) as follows:*

- $\text{ABE}_2.\text{Setup}(1^\kappa)$: Takes as input a security parameter 1^κ and outputs a public master key fmpk and a master secret key fmsk .
- $\text{ABE}_2.\text{KeyGen}(\text{fmsk}, P)$: Given a master secret key fmsk and a predicate $P \in \mathcal{P}$, outputs a key fsk_P corresponding to P .
- $\text{ABE}_2.\text{Enc}(\text{fmpk}, x, M_0, M_1)$: Takes as input the public key fmpk , an attribute $x \in \{0, 1\}^n$, for some n , and two messages $M_0, M_1 \in \mathcal{M}$ and outputs a ciphertext c .
- $\text{ABE}_2.\text{Dec}(\text{fsk}_P, c)$: Takes as input a secret key for a predicate and a ciphertext and outputs $M^* \in \mathcal{M}$.

Correctness. For any polynomial $n(\cdot)$, for every sufficiently large security parameter κ , if $n = n(\kappa)$, for all predicates $P \in \mathcal{P}_n$, attributes $x \in \{0, 1\}^n$, messages $M_0, M_1 \in \mathcal{M}$:

$$\Pr \left[\begin{array}{l} (\text{fmpk}, \text{fmsk}) \leftarrow \text{ABE}_2.\text{Setup}(1^\kappa); \\ \text{fsk}_P \leftarrow \text{ABE}_2.\text{KeyGen}(\text{fmsk}, P); \\ c \leftarrow \text{ABE}_2.\text{Enc}(\text{fmpk}, x, M_0, M_1); \\ M^* \leftarrow \text{ABE}_2.\text{Dec}(\text{fsk}_P, c); \\ M^* = M_{P(x)} \end{array} \right] = 1 - \text{negl}(\kappa).$$

We now define the security for single-key two-outcome attribute-based encryption. Intuitively, the security definition requires that, using a token for a predicate P , an adversary can decrypt one of the two messages encrypted in C based on the evaluation of P on the attribute, but does not learn anything about the other message.

Definition 2.11 (Two-outcome attribute-based encryption security). *Let ABE_2 be a two-outcome attribute-based encryption scheme for the class of predicates $\mathcal{P} = \{P_n\}_{n \in \mathbb{N}}$ and associated message space \mathcal{M} and let $A = (A_1, A_2, A_3)$ be a triple of p.p.t. adversaries. Consider the following experiment.*

$\text{Exp}_{\text{ABE}_2}(1^\kappa)$:

- 1: $(\text{fmpk}, \text{fmsk}) \leftarrow \text{ABE}_2.\text{Setup}(1^\kappa)$
- 2: $(P, \text{state}_1) \leftarrow A_1(\text{fmpk})$
- 3: $\text{sk}_P \leftarrow \text{ABE}_2.\text{KeyGen}(\text{fmsk}, P)$
- 4: $(M, M_0, M_1, x, \text{state}_2) \leftarrow A_2(\text{state}_1, \text{sk}_P)$
- 5: Choose a bit b at random. Then, let

$$c = \begin{cases} \text{ABE}_2.\text{Enc}(\text{fmpk}, x, M, M_b), & \text{if } P(x) = 0, \\ \text{ABE}_2.\text{Enc}(\text{fmpk}, x, M_b, M), & \text{otherwise.} \end{cases}$$

- 6: $b' \leftarrow A_3(\text{state}_2, c)$. If $b = b'$, $\exists n$ such that, for all $P \in \mathcal{P}_n$, messages $M, M_0, M_1 \in \mathcal{M}$, $|M_0| = |M_1|$, $x \in \{0, 1\}^n$, output 1, else output 0.
-

We say that the scheme is a fully-secure single-key two-outcome ABE if for all p.p.t. adversaries A , and for all sufficiently large security parameters κ :

$$\Pr[\text{Exp}_{\text{ABE}_2, A}(1^\kappa) = 1] \leq 1/2 + \text{negl}(\kappa).$$

The scheme is single-key selectively secure if A needs to provide x before receiving fmpk .

As before, we need only a single-key ABE_2 scheme for our construction.

A class of predicates $\{\mathcal{P}_n\}_n$ is closed under negation if for all input sizes n and for all predicates $p \in \mathcal{P}_n$, we have $\bar{p} \in \mathcal{P}_n$; \bar{p} is the negation of p , namely $\bar{p}(y) = 1 - p(y)$ for all y .

Claim 2.5. Assuming there is an ABE scheme for a class of predicates closed under negation, there exists a two-outcome ABE scheme for the same class of predicates.

The proof of this claim is immediate and we present it in Appendix B, for completeness.

2.6 Functional Encryption (FE)

We recall the functional encryption definition from the literature [KSW08, BSW, GVW12] with some notational changes.

Definition 2.12 (Functional Encryption). A functional encryption scheme FE for a class of functions $\mathcal{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$ represented as boolean circuits with an n -bit input, is a tuple of four p.p.t. algorithms (FE.Setup, FE.KeyGen, FE.Enc, FE.Dec) such that:

- FE.Setup(1^κ) takes as input the security parameter 1^κ and outputs a master public key fmpk and a master secret key fmsk .
- FE.KeyGen(fmsk, f) takes as input the master secret key fmsk and a function $f \in \mathcal{F}$ and outputs a key fsk_f .
- FE.Enc(fmpk, x) takes as input the master public key fmpk and an input $x \in \{0, 1\}^*$ and outputs a ciphertext c .
- FE.Dec(fsk_f, c) takes as input a key fsk_f and a ciphertext c and outputs a value y .

Correctness. For any polynomial $n(\cdot)$, for every sufficiently large security parameter κ , for $n = n(\kappa)$, for all $f \in \mathcal{F}_n$, and all $x \in \{0, 1\}^n$,

$$\Pr[(\text{fmpk}, \text{fmsk}) \leftarrow \text{FE.Setup}(1^\kappa); \text{fsk}_f \leftarrow \text{FE.KeyGen}(\text{fmsk}, f); c \leftarrow \text{FE.Enc}(\text{fmpk}, x) : \text{FE.Dec}(\text{fsk}_f, c) = f(x)] = 1 - \text{negl}(\kappa).$$

2.6.1 Security of Functional Encryption

Intuitively, the security of functional encryption requires that an adversary should not learn anything about the input x other than the computation result $C(x)$, for some circuit C for which a key was issued (the adversary can learn the circuit C). As mentioned, two notions of security have been used in previous work: full and selective security, with the same meaning as for ABE. We present both definitions because we achieve them with different parameters of the gapSVP assumption. Our definitions are simulation-based: the security definition states that whatever information an adversary is able to learn from the ciphertext and the function keys can be simulated given only the function keys and the output of the function on the inputs.

Definition 2.13. (FULL-SIM-Security) Let FE be a functional encryption scheme for the family of functions $\mathcal{F} = \{\mathcal{F}_n\}_{n \in \mathbb{N}}$. For every p.p.t. adversary $A = (A_1, A_2)$ and p.p.t. simulator S , consider the following two experiments:

$\text{Exp}_{\text{FE}, A}^{\text{real}}(1^\kappa):$	$\text{Exp}_{\text{FE}, A, S}^{\text{ideal}}(1^\kappa):$
1: $(\text{fmpk}, \text{fmsk}) \leftarrow \text{FE.Setup}(1^\kappa)$	1: $(\text{fmpk}, \text{fmsk}) \leftarrow \text{FE.Setup}(1^\kappa)$
2: $(f, \text{state}_A) \leftarrow A_1(\text{fmpk})$	2: $(f, \text{state}_A) \leftarrow A_1(\text{fmpk})$
3: $\text{fsk}_f \leftarrow \text{FE.KeyGen}(\text{fmsk}, f)$	3: $\text{fsk}_f \leftarrow \text{FE.KeyGen}(\text{fmsk}, f)$
4: $(x, \text{state}'_A) \leftarrow A_2(\text{state}_A, \text{fsk}_f)$	4: $(x, \text{state}'_A) \leftarrow A_2(\text{state}_A, \text{fsk}_f)$
5: $c \leftarrow \text{FE.Enc}(\text{fmpk}, x)$	5: $\tilde{c} \leftarrow S(\text{fmpk}, \text{fsk}_f, f, f(x), 1^{ x })$
6: Output (state'_A, c)	6: Output $(\text{state}'_A, \tilde{c})$

The scheme is said to be (single-key) FULL-SIM-secure if there exists a p.p.t. simulator S such that for all pairs of p.p.t. adversaries (A_1, A_2) , the outcomes of the two experiments are computationally indistinguishable:

$$\left\{ \text{Exp}_{\text{FE}, A}^{\text{real}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx} \left\{ \text{Exp}_{\text{FE}, A, S}^{\text{ideal}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}}.$$

We now define selective security, which is a weakening of full security, by requiring the adversary to provide the challenge input x before seeing the public key or any other information besides the security parameter. We simply specify the difference from full security.

Definition 2.14 (SEL-SIM-Security). The same as Def. 2.13, but modify the game so that the first step consists of A specifying the challenge input x given only the security parameter.

It is easy to see that the full simulation definition (FULL-SIM-security) implies the selective definition (SEL-SIM-security).

The literature [BSW, AGVW12] has considered another classification for simulation-based definitions: adaptive versus non-adaptive security. In the adaptive case, the adversary is allowed to ask for a function f after seeing the ciphertext c for an input x . In the non-adaptive case, the adversary must first provide f

and only then ask for encryptions of inputs x . Our definition falls in the non-adaptive category. Boneh et al. [BSW] have shown that adaptive simulation-based security is unachievable even for single-key functional encryption for the simple functionality of identity-based encryption. As such, the adaptive definition appears too strong and is unachievable for general functionalities, so we use non-adaptive security.

Remark 2.6. *Attribute-based encryption can be viewed as functional encryption for a specific class of functionalities, where the additional information leaked is part of the output to the function. Namely, consider a class of functions \mathcal{F} whose plaintext space consists of pairs of values from $\{0, 1\}^n \times \mathcal{M}$, where $\{0, 1\}^n$ is the attribute space (with an attribute size of n) and \mathcal{M} is the message space. The class of functions for ABE is more specific: there exists an associated predicate class $\mathcal{P} = \{\mathcal{P}_n\}_{n \in \mathbb{N}}$ to \mathcal{F} such that for every n , for every $f \in \mathcal{F}_n$, there is an associated predicate $P \in \mathcal{P}_n$ to f such that*

$$f(x, M) = \begin{cases} (x, M), & \text{if } P(x) = 1, \\ (x, \perp), & \text{otherwise.} \end{cases}$$

Since the attribute x is in the output of the function no matter what P is, x leaks from the scheme no matter what $(x$ is public). Therefore, this functionality leads to weaker security guarantees than functional encryption in a conceptual way: the value to be computed on, x , leaks with ABE (whereas the value M on which P does not compute remains secret when $P(x) = 0$), whereas the input x to the computation is hidden with FE.

3 Our Functional Encryption Scheme

In this section, we present our main result: the construction of a functional encryption scheme FE. We refer the reader to the introduction (Sec. 1.2) for an overview of our approach, and we proceed directly with the construction here.

We use three building blocks in our construction: a (leveled) fully homomorphic encryption scheme FHE, a (leveled) two-outcome attribute-based encryption scheme ABE₂, and a Yao garbling scheme Gb.

We let $\text{FHE.Eval}_f(\text{hpk}, \bar{\psi})$ denote the circuit that performs homomorphic evaluation of the function f on the vector of ciphertexts $\bar{\psi} := (\psi_1, \dots, \psi_n)$ using the public key hpk , and we will let $\text{FHE.Eval}_f^i(\text{hpk}, \psi)$ denote the predicate that computes the i -th output bit of $\text{FHE.Eval}_f(\text{hpk}, \bar{\psi})$. Namely,

$$\text{FHE.Eval}_f(\text{hpk}, \bar{\psi}) = \left(\text{FHE.Eval}_f^1(\text{hpk}, \bar{\psi}), \dots, \text{FHE.Eval}_f^\lambda(\text{hpk}, \bar{\psi}) \right),$$

where $\lambda = \lambda(\kappa) = |\text{FHE.Eval}_f(\text{hpk}, \bar{\psi})|$. Our main theorem then says:

Theorem 3.1. *There is a (fully/selectively secure) single-key functional encryption scheme $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$ for any class of circuits \mathcal{C} that take n bits of input and produce a one-bit output, assuming the existence of the following primitives:*

- *an IND-CPA-secure \mathcal{C} -homomorphic encryption scheme $\text{FHE} = (\text{FHE.KeyGen}, \text{FHE.Enc}, \text{FHE.Eval}, \text{FHE.Dec})$;*
- *a (fully/selectively secure) single-key attribute-based encryption scheme $\text{ABE} = (\text{ABE.Setup}, \text{ABE.KeyGen}, \text{ABE.Enc}, \text{ABE.Dec})$ for the class of predicates $\mathcal{P} = \mathcal{P}_{\mathcal{C}, \text{FHE}}$ where*

$$\mathcal{P}_{\mathcal{C}, \text{FHE}} = \{\text{FHE.Eval}_C^i, 1 - \text{FHE.Eval}_C^i : C \in \mathcal{C} \text{ and } i \in \{1, \dots, \lambda\}\}; \quad \text{and}$$

- a Yao garbling scheme $Gb = (Gb.Garble, Gb.Enc, Gb.Eval)$ that is input- and circuit-private.

The succinctness property of the functional encryption scheme is summarized as follows: the size of the ciphertexts $ctsize_{FE}(n)$ in the resulting scheme for n bits of input is

$$2 \cdot ctsize_{FHE} \cdot [ctsize_{ABE}(n \cdot ctsize_{FHE} + psize_{FHE})] + \text{poly}(\kappa, ctsize_{FHE}, sksize_{FHE}).$$

where $ctsize_{ABE}(k)$ denotes the size of the ciphertexts in the attribute-based encryption scheme for a k -bit attribute and a $\text{poly}(\kappa)$ -bit message, $ctsize_{FHE}$ denotes the size of the ciphertexts in the fully homomorphic encryption scheme for a single-bit message and $psize_{FHE}$ (resp. $sksize_{FHE}$) denotes the size of the public key (resp. secret key) in the fully homomorphic encryption scheme.

Since garbling schemes can be constructed from one-way functions, our theorem says that we can move from attribute-based encryption, in which the part of the input that the function computes on leaks, to a functional encryption scheme, in which no part of the input leaks using fully homomorphic encryption and Yao garbled circuits.

We can see that if the ciphertext size in the ABE scheme and the fully homomorphic encryption scheme does not depend on the circuit size (and thus, those schemes are by themselves succinct), then neither will the resulting ciphertexts of the FE scheme depend on the circuit size; namely, the reduction does not blow up the ciphertexts and is “succinctness-preserving”. We know of both a leveled FHE scheme and a leveled ABE scheme ([GVW13]) with ciphertext lengths independent of the size of the circuits to evaluate; the ciphertext size in these schemes just depends on the depth of the circuits.

We note that fully homomorphic encryption schemes with succinct ciphertexts that are also independent of depth are known, albeit under the stronger assumption of circular security of the underlying schemes. Thus, if the result of [GVW13] can be improved to remove the depth dependency of the ciphertexts in the ABE scheme, one automatically obtains a corresponding result for ABE using our reduction.

Our theorem needs the ABE scheme to be secure only with a single key, even though the recent constructions [GVW13] and [SW12] can tolerate an arbitrary number of keys.

Our main theorem is thus a reduction, which has a number of useful corollaries. The first and perhaps the most important one shows how to combine the leveled fully homomorphic encryption scheme from [BV11a, BGV12] with the recent construction of a leveled attribute-based encryption scheme from [GVW13] to obtain a leveled functional encryption scheme based solely on the hardness of LWE. In other words, the corollary says that for every depth d , there is a functional encryption scheme for the class of all Boolean circuits of (arbitrary) polynomial size and depth at most d . The size of the ciphertexts in the scheme grows with d , and is of course independent of the size of the circuits it supports.

Let d and p be polynomial functions. Define $\mathcal{C}_{n,d(n),p(n)}$ to be the class of all Boolean circuits on n inputs of depth at most $d(n)$ and size at most $p(n)$. Let $\mathcal{C}_{n,d(n)} := \bigcup_{\text{polynomial } p} \mathcal{C}_{n,d(n),p(n)}$.

Corollary 3.2 (The LWE Instantiation). *We have the following two constructions of functional encryption based on the worst-case hardness of lattice problems:*

- Assume that there is a constant $0 < \epsilon < 1$ such that for every sufficiently large ℓ , the approximate shortest vector problem gapSVP in ℓ dimensions is hard to approximate to within a $2^{O(\ell^\epsilon)}$ factor (in polynomial time) in the worst case. Then, for every n and every polynomial $d = d(n)$, there is a selectively-secure (succinct single-key) functional encryption scheme for the class $\mathcal{C}_{n,d(n)}$ where encrypting n bits produces ciphertexts of length $\text{poly}(n, \kappa, d^{1/\epsilon})$.

- Assume that there is a constant $0 < \epsilon < 1$ such that for every sufficiently large ℓ , the approximate shortest vector problem gapSVP in ℓ dimensions is hard to approximate to within a $2^{O(\ell^\epsilon)}$ factor in time $2^{O(\ell^\epsilon)}$ in the worst case. Then, for every n and every polynomial $d = d(n)$, there is a fully-secure (succinct single-key) functional encryption scheme for the class $\mathcal{C}_{n,d(n)}$ where encrypting n bits produces ciphertexts of length $\text{poly}(n^{1/\epsilon}, \kappa, d^{1/\epsilon^2})$.

The corollary follows directly from Theorem 3.1, by invoking the leveled fully homomorphic encryption scheme of [BV11a] (see Theorem 2.1) and the leveled attribute-based encryption scheme of [GVW13] (see Theorem 2.4). The concrete constructions and proofs in fact go through the learning with errors (LWE) problem; we refer to [BV11a, GVW13] for the concrete setting of parameters.

Letting *universal* attribute-based encryption or functional encryption denote a single attribute-based encryption or functional encryption scheme, respectively, that supports the class of all polynomial-size circuits, we have the following corollary:

Corollary 3.3 (Universal Functional Encryption). *Assuming that fully homomorphic encryption schemes exist and universal single-key attribute-based encryption schemes exist, there is a universal single-key functional encryption scheme.*

Of the two prerequisites mentioned above, we know that fully homomorphic encryption schemes exist (albeit under stronger assumptions than merely LWE). Thus, the corollary provides a way to immediately translate any universal attribute-based encryption scheme into a functional encryption scheme. We point out that universal functional encryption schemes, by definition, have succinct ciphertexts.

A recent result of Gorbunov, Vaikuntanathan and Wee [GVW12] shows how to generically convert single-key functional encryption schemes into q -keys functional encryption schemes for any bounded q , where the latter provide security against an attacker that can obtain secret keys of up to q functions of her choice. The size of the ciphertexts in the q -keys scheme grows polynomially with q .

Corollary 3.4 (Many queries, using [GVW12]). *For every $q = q(\kappa)$, there is a (fully/selectively-secure) q -keys succinct functional encryption scheme for any class of circuits \mathcal{C} that take n bits of input and produce a one-bit output, assuming the existence of primitives as in Theorem 3.1. The size of the ciphertexts $\text{ctsize}_{\text{FE}}(n)$ in the resulting scheme is q times as large as in Theorem 3.1.*

Finally, a functional encryption scheme for circuits that output multiple bits can be constructed by thinking of the circuit as many circuits each with one-bit output, and modifying the key generation procedure to produce keys for each of them. This gives us the following corollary although we remark that more efficient methods of achieving this directly are possible using homomorphic encryption schemes that pack multiple bits into a single ciphertext [SV11, BGV12, GHS12a].

Corollary 3.5 (Many queries, many output bits). *For every $q = q(\kappa)$ and $k = k(n)$, there is a (fully/selectively secure) q -keys functional encryption scheme for any class of circuits \mathcal{C} that take n bits of input and produce k bits of output, assuming the existence of primitives as in Theorem 3.1. The size of the ciphertexts $\text{ctsize}_{\text{FE}}(n)$ in the resulting scheme is qk times as large as in Theorem 3.1.*

Remark 3.6 (On the necessity of single-key security). *We note that even though the work of [GVW13] provides an attribute-based scheme that is secure even if the adversary obtains secret keys for polynomially many functions, our theorem gives us only a single-key secure scheme. Indeed, this is inherent by the impossibility result of [AGVW12] if we ask for (even a very weak notion of) simulation security, as we do here. Corollary 3.4 gives us a way to get (simulation-)security with q queries for any a priori bounded q , albeit at the expense of the ciphertext growing as a function of q .*

Remark 3.7 (On composing our functional encryption scheme). *One might wonder if chaining is possible with our FE scheme. Namely, one could try to generate keys for a function f that computes another function f_1 on an input x and then outputs $f_1(x)$ together with a new encryption of x under a different public key for the FE scheme. The new encryption of x could be used to compute a second function $f_2(x)$ and an encryption of x under yet another public key. This chain could potentially repeat and its benefit is that it allows us to compute multiple functions on x (and overcome the single-key property). However, this approach allows only a very small number of iterations because, in order to produce one bit of output from FE.Dec, the ciphertexts output by FE.Enc are polynomial in κ . To obtain an FE ciphertext as result of FE.Dec, one needs to have started with ciphertexts of size quadratic in the first polynomial. If we want to chain the scheme q times, the original ciphertext must have been exponential in q .*

3.1 Construction

For simplicity, we construct FE for functions outputting one bit; functions with larger outputs can be handled by repeating our scheme below for every output bit.

From Claim 2.5, the existence of a secure single-key ABE scheme implies the existence of a two-outcome single-key ABE scheme, which we denote ABE_2 . Let $\lambda = \lambda(\kappa)$ be the length of the ciphertexts in the FHE scheme (both from encryption and evaluation). The construction of $\text{FE} = (\text{FE.Setup}, \text{FE.KeyGen}, \text{FE.Enc}, \text{FE.Dec})$ proceeds as follows.

Setup $\text{FE.Setup}(1^\kappa)$: Run the setup algorithm for the two-outcome ABE scheme λ times:

$$(\text{fmpk}_i, \text{fmsk}_i) \leftarrow \text{ABE}_2.\text{Setup}(1^\kappa) \text{ for } i \in [\lambda].$$

Output as master public key and secret key:

$$\text{MPK} = (\text{fmpk}_1, \dots, \text{fmpk}_\lambda) \text{ and } \text{MSK} = (\text{fmsk}_1, \dots, \text{fmsk}_\lambda).$$

Key Generation $\text{FE.KeyGen}(\text{MSK}, f)$: Let n be the number of bits in the input to the circuit f . If hpk is an FHE public key and ψ_1, \dots, ψ_n are FHE ciphertexts, recall that $\text{FHE.Eval}_f^i(\text{hpk}, \psi_1, \dots, \psi_n)$ is the i -th bit of the homomorphic evaluation of f on ψ_1, \dots, ψ_n ($\text{FHE.Eval}(\text{hpk}, f, \psi_1, \dots, \psi_n)$), where $i \in [\lambda]$. Thus, $\text{FHE.Eval}_f^i : \{0, 1\}^{|\text{hpk}|} \times \{0, 1\}^{n\lambda} \rightarrow \{0, 1\}$.

1. Run the key generation algorithm of ABE_2 for the functions FHE.Eval_f^i (under the different master secret keys) to construct secret keys:

$$\text{fsk}_i \leftarrow \text{ABE}_2.\text{KeyGen}(\text{fmsk}_i, \text{FHE.Eval}_f^i) \text{ for } i \in [\lambda].$$

2. Output the tuple $\text{fsk}_f := (\text{fsk}_1, \dots, \text{fsk}_\lambda)$ as the secret key for the function f .

Encryption $\text{FE.Enc}(\text{MPK}, x)$: Let n be the number of bits of x , namely $x = x_1 \dots x_n$. Encryption proceeds in three steps.

1. Generate a fresh key pair $(\text{hpk}, \text{hsk}) \leftarrow \text{FHE.KeyGen}(1^\kappa)$ for the (leveled) fully homomorphic encryption scheme. Encrypt each bit of x homomorphically: $\psi_i \leftarrow \text{FHE.Enc}(\text{hpk}, x_i)$. Let $\psi := (\psi_1, \dots, \psi_n)$ be the encryption of the input x .

2. Run the Yao garbled circuit generation algorithm to produce a garbled circuit for the FHE decryption algorithm $\text{FHE.Dec}(\text{hsk}, \cdot) : \{0, 1\}^\lambda \rightarrow \{0, 1\}$ together with 2λ labels L_i^b for $i \in [\lambda]$ and $b \in \{0, 1\}$. Namely,

$$\left(\Gamma, \{L_i^0, L_i^1\}_{i=1}^\lambda \right) \leftarrow \text{Gb.Garble}(1^\kappa, \text{FHE.Dec}(\text{hsk}, \cdot)),$$

where Γ is the garbled circuit and the L_i^b are the input labels.

3. Produce encryptions c_1, \dots, c_λ using the ABE_2 scheme:

$$c_i \leftarrow \text{ABE}_2.\text{Enc}(\text{fmpk}_i, (\text{hpk}, \psi), L_i^0, L_i^1) \text{ for } i \in [\lambda],$$

where (hpk, ψ) comes from the first step, and the labels (L_i^0, L_i^1) come from the second step.

4. Output the ciphertext $c = (c_1, \dots, c_\lambda, \Gamma)$.

Decryption $\text{FE.Dec}(\text{fsk}_f, c)$:

1. Run the ABE_2 decryption algorithm on the ciphertexts c_1, \dots, c_λ to recover the labels for the garbled circuit. In particular, let

$$L_i^{d_i} \leftarrow \text{ABE}_2.\text{Dec}(\text{fsk}_i, c_i) \text{ for } i \in [\lambda],$$

where d_i is equal to $\text{FHE.Eval}_f^i(\text{hpk}, \psi)$.

2. Now, armed with the garbled circuit Γ and the labels $L_i^{d_i}$, run the garbled circuit evaluation algorithm to compute

$$\text{Gb.Eval}(\Gamma, L_1^{d_1}, \dots, L_\lambda^{d_\lambda}) = \text{FHE.Dec}(\text{hsk}, d_1 d_2 \dots d_\lambda) = f(x).$$

3.2 Proof

We now proceed to prove Theorem 3.1 by proving that the theorem holds for our construction above.

Proof of Theorem 3.1. We first argue correctness.

Claim 3.8. *The above scheme is a correct functional encryption scheme (Def. 2.12).*

Proof. Let us examine the values we obtain in $\text{FE.Dec}(\text{fsk}_f, c_1, \dots, c_\lambda, \Gamma)$. In Step (1), by the correctness of the ABE_2 scheme used, d_i is the i -th bit of $\text{FHE.Eval}_f(\text{hpk}, \psi)$.

Therefore, the inputs to the garbled circuit Γ in Step (2) are the labels corresponding to $\text{FHE.Eval}_f(\text{hpk}, \psi)$. By the correctness of the FHE scheme, decrypting $\text{FHE.Eval}_f(\text{hpk}, \psi)$ results in $f(x)$. Finally, by the correctness of the garbling scheme, the FHE ciphertext gets decrypted correctly, yielding $f(x)$ as the output of FE.Dec . \square

We now prove the succinctness property which follows directly from our construction. The output of FE.Enc consists of λ ABE_2 ciphertexts and a garbled circuit. First, λ equals $\text{ctsize}_{\text{FHE}}$. Second, each ABE_2 ciphertext consists of two ABE ciphertexts generated by ABE.Enc on input $n\text{ctsize}_{\text{FHE}} + \text{pksize}_{\text{FHE}}$ bits. The labels of the garbled circuit are $\text{poly}(\kappa)$ in size. Third, the garbled circuit is the output of Gb.Garble so its size is polynomial in the size of the input circuit, which in turn is polynomial in $\text{sksize}_{\text{FHE}}$ and $\text{ctsize}_{\text{FHE}}$. Therefore, overall, we obtain $2\text{ctsize}_{\text{FHE}} \cdot \text{ctsize}_{\text{ABE}}(n\text{ctsize}_{\text{FHE}} + \text{pksize}_{\text{FHE}}) + \text{poly}(\kappa, \text{sksize}_{\text{FHE}}, \text{ctsize}_{\text{FHE}})$.

We can thus see that if FHE and ABE produce ciphertexts independent of the circuit size, then so will our functional encryption scheme.

We focus on the full security case: namely, assuming ABE_2 is fully secure, we show that the resulting FE scheme is fully secure. We then discuss the proof for the selective case.

For full security, we construct a p.p.t. simulator S that achieves Def. 2.13. S receives as input $(\text{MPK}, \text{fsk}_f, f, f(x), 1^n)$ and must output \tilde{c} such that the real and ideal experiments in Def. 2.13 are computationally indistinguishable. Intuitively, S runs a modified version of FE.Enc to mask the fact that it does not know x .

Simulator S on input $(\text{MPK}, \text{fsk}_f, f, f(x), 1^n)$:

1. Choose a key pair $(\text{hpk}, \text{hsk}) \leftarrow \text{FHE.KeyGen}(1^\kappa)$ for the homomorphic encryption scheme (where S can derive the security parameter κ from the sizes of the inputs it gets). Encrypt 0^n (n zero bits) with FHE by encrypting each bit individually and denote the ciphertext $\hat{0} := (\hat{0}_1 \leftarrow \text{FHE.Enc}(\text{hpk}, 0), \dots, \hat{0}_n \leftarrow \text{FHE.Enc}(\text{hpk}, 0))$.
2. Let $\text{Sim}_{\text{Garble}}$ be the simulator for the Yao garbling scheme (described in Def. 2.7) for the class of circuits corresponding to $\text{FHE.Dec}(\text{hsk}, \cdot)$. Run $\text{Sim}_{\text{Garble}}$ to produce a simulated garbled circuit $\tilde{\Gamma}$ for the FHE decryption algorithm $\text{FHE.Dec}(\text{hsk}, \cdot) : \{0, 1\}^\lambda \rightarrow \{0, 1\}$ together with the simulated encoding consisting of one set of λ labels \tilde{L}_i for $i = 1 \dots \lambda$. Namely,

$$\left(\tilde{\Gamma}, \{\tilde{L}_i\}_{i=1}^\lambda \right) \leftarrow \text{Sim}_{\text{Garble}}(1^\kappa, f(x), 1^{|\text{FHE.Dec}(\text{hsk}, \cdot)|}, 1^\lambda).$$

The simulator S can invoke $\text{Sim}_{\text{Garble}}$ because it knows $f(x)$, and can compute the size of the $\text{FHE.Dec}(\text{hsk}, \cdot)$ circuit, and λ from the sizes of the input parameters.

3. Produce encryptions $\tilde{c}_1, \dots, \tilde{c}_\lambda$ under the ABE_2 scheme in the following way. Let

$$\tilde{c}_i \leftarrow \text{ABE}_2.\text{Enc} \left(\text{fmpk}_i, (\text{hpk}, \hat{0}), \tilde{L}_i, \tilde{L}_i \right),$$

where S uses each simulated label \tilde{L}_i twice.

4. Output $\tilde{c} = (\tilde{c}_1, \dots, \tilde{c}_\lambda, \tilde{\Gamma})$.

To prove indistinguishability of the real and ideal experiments (Def. 2.13), we define a sequence of hybrid experiments, and then invoke the security definitions of the underlying schemes (FHE, garbled circuit, and ABE_2 respectively) to show that the outcome of the hybrid experiments are computationally indistinguishable.

Hybrid 0 is the output of the ideal experiment from Def. 2.13 for our FE construction with simulator S . We denote it $\text{Exp}_{\text{FE}, A}^{H_0} (= \text{Exp}_{\text{FE}, A, S}^{\text{ideal}})$.

Hybrid 1 ($\text{Exp}_{\text{FE}, A}^{H_1}$) is the same as Hybrid 0, except that the simulated ciphertext for Hybrid 1 (which we denote $\tilde{c}^{(1)}$), changes. Let $\tilde{c}^{(1)}$ be the ciphertext obtained by running the algorithm of S , except that in Step (3), encrypt x instead of 0, namely:

$$\tilde{c}_i^{(1)} \leftarrow \text{ABE}_2.\text{Enc} \left(\text{fmpk}_i, (\text{hpk}, \psi), \tilde{L}_i, \tilde{L}_i \right),$$

where $\psi \leftarrow (\text{FHE.Enc}(\text{hpk}, x_1), \dots, \text{FHE.Enc}(\text{hpk}, x_n))$. Let

$$\tilde{c}^{(1)} = (\tilde{c}_1^{(1)}, \dots, \tilde{c}_\lambda^{(1)}, \tilde{\Gamma}).$$

Hybrid 2 ($\text{Exp}_{\text{FE},A}^{H_2}$) is the same as Hybrid 1, except that in Step (2), the ciphertext contains a real garbled circuit

$$\left(\Gamma, \{L_i^0, L_i^1\}_{i=1}^\lambda \right) \leftarrow \text{Gb.Garble}(\text{FHE.Dec}(\text{hsk}, \cdot)).$$

Let $d_i = \text{FHE.Eval}_f^i(\text{hpk}, \psi)$. In Step (3), include $L_i^{d_i}$ twice in the ABE encryption; namely:

$$\tilde{c}_i^{(2)} \leftarrow \text{ABE}_2.\text{Enc} \left(\text{fmpk}_i, (\text{hpk}, \psi), L_i^{d_i}, L_i^{d_i} \right), \text{ and}$$

$$\tilde{c}^{(2)} = (\tilde{c}_1^{(2)}, \dots, \tilde{c}_\lambda^{(2)}, \Gamma).$$

Hybrid 3 ($\text{Exp}_{\text{FE},A}^{H_3}$) is the output of the real experiment from Def. 2.13 for our FE construction.

We prove each pair of consecutive hybrids to be computationally indistinguishable in the following three lemmas, Lemmas 3.9, 3.10, and 3.11.

Lemma 3.9. *Assuming FHE is IND-CPA-secure, Hybrid 0 and Hybrid 1 are computationally indistinguishable.*

Proof. We proceed by contradiction. We assume that there exist p.p.t. adversaries $A = (A_1, A_2)$ and a p.p.t. distinguisher D such that D (with A) can distinguish between Hybrid 0 and Hybrid 1 above. Namely, there exists a polynomial $p(\cdot)$ such that, for infinitely many κ ,

$$|\Pr[D(\text{Exp}_{\text{FE},A}^{H_0}(1^\kappa)) = 1] - \Pr[D(\text{Exp}_{\text{FE},A}^{H_1}(1^\kappa)) = 1]| \geq 1/p(\kappa). \quad (1)$$

We construct a p.p.t. adversary $R = (R_1, R_2)$ that can break the semantic security of FHE. Adversary R_1 outputs an n -bit value x for some n , and adversary R_2 receives as input either homomorphic encryption of x or of 0^n , and it will distinguish between these two. Distinguishing successfully implies that there is an adversary that can distinguish successfully in Def. 2.5, by a standard hybrid argument.

To determine x , adversary R_1 works as follows:

1. Run $\text{Exp}_{\text{FE},A,S}^{\text{ideal}}(1^\kappa)$ (Def. 2.13) from Step (1) to Step (4) and let x be the output of A_2 in Step (4).
2. Output x .

To distinguish between encryption of x or 0^n , adversary R_2 receives input hpk^* , the FHE public key, and an encryption E^* of x or 0^n and works as follows:

1. Run a modified algorithm of S by using hpk^* instead of generating fresh FHE keys and using E^* instead of encrypting 0^n . Namely:
 - (a) Generate $(\tilde{\Gamma}, \{\tilde{L}_i\}_{i=1}^\lambda)$ as in Step (2) of S .
 - (b) Output $c^* = (c_1^*, \dots, c_\lambda^*)$ for $c_i^* = \text{ABE}_2.\text{Enc}(\text{fmpk}_i, ((\text{hpk}^*, E^*), \tilde{L}_i, \tilde{L}_i))$.
2. Feed $(c^*, \tilde{\Gamma})$ to D and output the decision of D .

Notice that if E^* is encryption of 0^n , R_2 simulates Hybrid 0 perfectly; when E^* is encryption of x , R_2 simulates Hybrid 1 perfectly. Therefore, D must have a probability of distinguishing between the two cases of

at least $1/p(\kappa)$ (Eq. (1)); moreover, whenever D distinguishes correctly, R also outputs the correct decision. Therefore:

$$\begin{aligned} & |\Pr[x \leftarrow R_1(1^\kappa); (\text{hsk}^*, \text{hpk}^*) \leftarrow \text{FHE.KeyGen}(1^\kappa) : R_2(\text{hpk}^*, \text{FHE.Enc}(\text{hpk}^*, x)) = 1] - \\ & \Pr[(\text{hsk}^*, \text{hpk}^*) \leftarrow \text{FHE.KeyGen}(1^\kappa) : R_2(\text{hpk}^*, \text{FHE.Enc}(\text{hpk}^*, 0^n)) = 1]| = \\ & |\Pr[D(\text{Exp}_{\text{FE},A}^{H_0}(1^\kappa)) = 1] - \Pr[D(\text{Exp}_{\text{FE},A}^{H_1}(1^\kappa)) = 1]| \geq 1/p(\kappa), \end{aligned}$$

which contradicts the IND-CPA security of the FHE scheme. \square

Lemma 3.10. *Assuming the garbled circuit is circuit- and input-private (Def. 2.7), Hybrid 1 and Hybrid 2 are computationally indistinguishable.*

Proof. We proceed by contradiction. Assume there exist p.p.t. adversaries $A = (A_1, A_2)$ and a p.p.t. distinguisher D such that D (with A) can distinguish Hybrid 1 and Hybrid 2 above. Namely, there exists a polynomial p such that, for infinitely many κ ,

$$|\Pr[D(\text{Exp}_{\text{FE},A}^{H_1}(1^\kappa)) = 1] - \Pr[D(\text{Exp}_{\text{FE},A}^{H_2}(1^\kappa)) = 1]| \geq 1/p(\kappa). \quad (2)$$

We construct a stateful p.p.t. adversary $R = (R.A, R.D)$ that can break the security of the garbling scheme from Def. 2.7. The adversary $R.A$ has to provide a circuit G and an input I and then $R.D$ needs to distinguish between the simulated and the real garbled circuits and input encodings.

The adversary $R.A$ computes I and G as follows.

1. Run Steps (1)–(4) from Def. 2.13, which are the same in Hybrid 1 and Hybrid 2 and obtain f from A_1 and x from A_2 .
2. Generate $(\text{hsk}, \text{hpk}) \leftarrow \text{FHE.KeyGen}(1^\kappa)$ and let $\psi \leftarrow \text{FHE.Enc}(\text{hpk}, x)$.
3. Output $G(\cdot) := \text{FHE.Dec}(\text{hsk}, \cdot)$ and $I := \text{FHE.Eval}_f(\text{hpk}, \psi)$ and the following state for $R.D$: $\alpha = (\psi, \text{fmpk}_i, \text{hpk})$.

The adversary $R.D$ receives as input a garbled circuit Γ^* and a set of labels, one for each i : $\{L_i^*\}_{i=1}^\lambda$. These could be outputs of either $\text{Sim}_{\text{Garble}}$ or of $\text{Gb.Garble}/\text{Gb.Enc}$ and $R.D$ decides which is an output of as follows:

1. Compute $c^* = (\{\text{ABE}_2.\text{Enc}(\text{fmpk}_i, ((\text{hpk}, \psi), L_i^*, L_i^*))\}_{i=1}^\lambda, \Gamma^*)$.
2. Run D on c^* and output what D outputs.

Notice that if $(\Gamma^*, \{L_i^*\}_{i=1}^\lambda)$ are outputs of $\text{Sim}_{\text{Garble}}$, R simulates Hybrid 1 perfectly; when $(\Gamma^*, \{L_i^*\}_{i=1}^\lambda)$ are outputs of the real garbling scheme, R simulates Hybrid 2 perfectly. Therefore, the probability that D distinguishes between the two cases at least is $1/p(\kappa)$ (Eq. (2)); moreover, whenever D distinguishes correctly, R also outputs the correct decision. Therefore:

$$\begin{aligned} & |\Pr[(G, I) \leftarrow R.A(1^\kappa) : R.D(\tilde{\Gamma}, \{\tilde{L}_i\}_{i=1}^\lambda) = 1] - \Pr[(G, I) \leftarrow R.A(1^\kappa) : R.D(\Gamma, \{L_i\}_{i=1}^\lambda) = 1]| = \\ & |\Pr[D(\text{Exp}_{\text{FE},A}^{H_1}(1^\kappa)) = 1] - \Pr[D(\text{Exp}_{\text{FE},A}^{H_2}(1^\kappa)) = 1]| \geq 1/p(\kappa), \end{aligned}$$

where, $(\tilde{\Gamma}, \{\tilde{L}_i\}_{i=1}^\lambda)$ are outputs of $\text{Sim}_{\text{Garble}}$ and $(\Gamma, \{L_i\}_{i=1}^\lambda)$ are outputs of $\text{Gb.Garble}/\text{Gb.Enc}$. This relation contradicts the security of the garbling scheme Def. 2.7. \square

Lemma 3.11. *Assuming the underlying ABE_2 scheme is fully secure, Hybrid 2 and Hybrid 3 in the fully secure setting above are computationally indistinguishable.*

Proof. In Hybrid 2 and Hybrid 3, there are λ ABE_2 encryptions, each with a pair of independent ABE_2 keys. First, we would like to prove that if Hybrid 2 and Hybrid 3 are computationally indistinguishable with only one of these encryptions, then they are computationally indistinguishable with λ encryptions. This would enable us to focus on only one ABE_2 ciphertext for the proof.

The argument proceeds in a standard way with a set of sub-hybrids, one for each index $i = 0 \dots \lambda$. The argument is straightforward because \tilde{c}_i and \tilde{c}_j (for $i \neq j$) use independently generated keys and the values encrypted with these keys are known to R . Hence, we present the hybrid argument briefly. Sub-hybrid 0 corresponds to Hybrid 2 and sub-hybrid λ corresponds to Hybrid 3. Sub-hybrid i has the first i ciphertexts as in Hybrid 2 and the rest $\lambda - i$ as in Hybrid 3.

If an adversary A can distinguish between sub-hybrids $i - 1$ and i , for some i , then he can distinguish Hybrid 2 and Hybrid 3 for only one pair of ciphertexts (c_i^2, c_i^3) ; the reason is that we can build an adversary B : B places the challenge ciphertext in slot i of the challenge to A and produces the ciphertexts for all other slots $j \neq i$ with the correct distribution; B can do so because these ciphertexts are encrypted with fresh ABE_2 keys and B has all the information it needs to generate them correctly.

Now we are left to prove that Hybrid 2 and Hybrid 3 are indistinguishable when there is only one ciphertext, say the ℓ -th ciphertext. Namely, we need to prove that:

$$\left\{ (\text{state}'_A, \tilde{c}_\ell^{(2)}) \leftarrow \text{Exp}_{\text{FE},A}^{H_2}(1^\kappa) \right\} \stackrel{c}{\approx} \left\{ (\text{state}'_A, \tilde{c}_\ell^{(3)}) \leftarrow \text{Exp}_{\text{FE},A}^{H_3}(1^\kappa) \right\}. \quad (3)$$

We prove this statement by contradiction. Assume there exist p.p.t. adversaries $A = (A_1, A_2)$ and distinguisher D that can distinguish the distributions in (3); namely, there exists a polynomial $p(\cdot)$ such that, for infinitely many κ ,

$$|\Pr[D(\text{Exp}_{\text{FE},A}^{H_2}(1^\kappa)) = 1] - \Pr[D(\text{Exp}_{\text{FE},A}^{H_3}(1^\kappa)) = 1]| \geq 1/p(\kappa). \quad (4)$$

We construct a p.p.t. adversary $R = (R_1, R_2, R_3)$ that breaks the security of ABE_2 from Def. 2.11. R_1 , R_2 and R_3 send state to each other as in Def. 2.11, but for simplicity we will not denote this explicitly. R_3 aims to guess b in this definition.

Intuition. A and D can distinguish between Hybrid 2 and Hybrid 3. The only difference between these hybrids is that \tilde{c}_ℓ contains encryption of $(L_\ell^{d_\ell}, L_\ell^{d_\ell})$ versus $(L_\ell^{d_\ell}, L_\ell^{1-d_\ell})$. However, the ABE_2 scheme does not decrypt $L_\ell^{1-d_\ell}$ by the definition of d_ℓ , so its security hides the value of $L_\ell^{1-d_\ell}$. Since A and D can distinguish between these hybrids, they must be breaking the security of ABE_2 . Therefore, R will use L_ℓ^ℓ and $L_\ell^{1-\ell}$ as part of its answers to C and then use D to distinguish its challenge.

Specifically, the adversary R_1 receives as input fmpk^* in Step 2 of Def. 2.11 and computes P as follows:

1. Interact with adversary A_1 by running Steps (1)–(2) from Defs. 2.13 as follows.
 - (a) Let $\text{fmpk}_\ell := \text{fmpk}^*$. Generate the rest of ABE_2 keys using the $\text{ABE}_2.\text{Setup}$ algorithm: $(\text{fmpk}_i, \text{fmsk}_i) \leftarrow \text{ABE}_2.\text{Setup}(1^\kappa)$ for $i \neq \ell$.
 - (b) Receive f from A_1 and output $P := \text{FHE.Eval}_f^\ell$.

Adversary R_2 receives sk_P^* in Step 4 of Def. 2.11 and computes M, M_0, M_1, x_c as follows:

1. Continue interaction with A_2 . To provide fsk_f to A_2 , compute $\text{fsk}_i \leftarrow \text{ABE}_2.\text{KeyGen}(\text{fmsk}_i, \text{FHE.Eval}_f^i)$ for $i \neq \ell$, and let $\text{fsk}_\ell := \text{sk}_P^*$.
2. Receive x from A_2 .
3. Run the real garbled circuit generation as in Hybrid 2 and 3. Let $L_\ell^{d_\ell}$ be defined as in Hybrid 2. Provide $M := L_\ell^{d_\ell}$, $M_0 := L_\ell^{d_\ell}$ and $M_1 := L_\ell^{1-d_\ell}$.
4. Let $x_c := (\text{hpk}, \psi)$ where $\psi \leftarrow (\text{FHE.Enc}(\text{hpk}, x_1), \dots, \text{FHE.Enc}(\text{hpk}, x_n))$, the bitwise FHE encryption of x .
5. Output (M, M_0, M_1, x_c) .

Adversary R_3 receives as input a challenge ciphertext c^* and decides if it corresponds to M_0 or to M_1 as follows:

1. Let $\tilde{c}_\ell := c^*$ and provide $(\text{state}'_A, \tilde{c}_\ell)$ to D .
2. Output D 's guess.

In order for D to distinguish (as in Eq. (4)), the input distribution to A must be the one from Hybrid 2 or 3. We can see that this is the case: if $b = 0$, R simulates Hybrid 2 perfectly, and if $b = 1$, R simulates Hybrid 3 perfectly. Moreover, whenever D distinguishes correctly, R also outputs the correct decision. Therefore, by a simple calculation, we can see that

$$\Pr[\text{Exp}_{\text{ABE}_2, R}(1^\kappa) = 1] \geq 1/2 + 1/2p(\kappa),$$

which contradicts the security of the ABE_2 scheme, Def. 2.11. \square

Returning to the proof of our theorem, by transitivity of computational indistinguishability, we showed that Hybrid 0 (the ideal experiment) is equivalent to Hybrid 3 (the real experiment), thus concluding our proof.

Selective security. The proof for the selective case follows similarly. The simulator S and the four hybrids are the same. Lemmas 3.9 and 3.10 proceed similarly, except that R now interacts with A as in the selective FE definition Def. 2.14 rather than Def. 2.13. The argument of Lemma 3.11 is the same, except that the order of some operations changes. This lemma makes the resulting FE scheme selective if one starts from a selective ABE_2 scheme. \square

4 Reusable Garbled Circuits

In this section, we show how to construct garbled circuits that can be reused; namely, a garbled circuit that can run on an arbitrary number of encoded inputs without compromising the privacy of the circuit or of the input. For this goal, we build on top of our functional encryption scheme.

The syntax and correctness of the reusable garbling schemes remains the same as the one for one-time garbling schemes (Def. 2.6). In Sec. 2.4, we provided the one-time security definition for circuit and input privacy, Def. 2.7. We begin by defining security for more than one-time usage.

Definition 4.1 (Input and circuit privacy with reusability). Let RgB be a garbling scheme for a family of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$. For a pair of p.p.t. algorithms $A = (A_1, A_2)$ and a p.p.t. simulator $S = (S_1, S_2)$, consider the following two experiments:

$\text{Exp}_{\text{RgB}, A}^{\text{real}}(1^\kappa):$	$\text{Exp}_{\text{RgB}, A, S}^{\text{ideal}}(1^\kappa):$
1: $(C, \text{state}_A) \leftarrow A_1(1^\kappa)$	1: $(C, \text{state}_A) \leftarrow A_1(1^\kappa)$
2: $(\text{gsk}, \Gamma) \leftarrow \text{RgB.Garble}(1^\kappa, C)$	2: $(\tilde{\Gamma}, \text{state}_S) \leftarrow S_1(1^\kappa, 1^{ C })$
3: $\alpha \leftarrow A_2^{\text{RgB.Enc}(\text{gsk}, \cdot)}(C, \Gamma, \text{state}_A)$	3: $\alpha \leftarrow A_2^{\text{O}(\cdot, C)[[\text{state}_S]]}(C, \tilde{\Gamma}, \text{state}_A)$
4: Output α	4: Output α

In the above, $\text{O}(\cdot, C)[[\text{state}_S]]$ is an oracle that on input x from A_2 , runs S_2 with inputs $C(x)$, $1^{|x|}$, and the latest state of S ; it returns the output of S_2 (storing the new simulator state for the next invocation).

We say that the garbling scheme RgB is input- and circuit-private with reusability if there exists a p.p.t. simulator S such that for all pairs of p.p.t. adversaries $A = (A_1, A_2)$, the following two distributions are computationally indistinguishable:

$$\left\{ \text{Exp}_{\text{RgB}, A}^{\text{real}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx} \left\{ \text{Exp}_{\text{RgB}, A, S}^{\text{ideal}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}}.$$

We can see that this security definition enables reusability of the garbled circuit: A_2 is allowed to make as many queries for input encodings as it wants.

From now on, by *reusable garbling scheme*, we will implicitly refer to a garbling scheme that has input and circuit privacy with reusability as in the definition above, Def. 4.1.

Remark 4.1. We can provide an alternate syntax for a reusable garbling scheme, and we can also construct a scheme with this syntax (and a similar security definition) from our functional encryption scheme. This syntax has an additional setup algorithm (separate from the garble algorithm) that produces the secret key necessary for encoding and for circuit garbling; such a syntax would allow the garbled circuit to be generated after the encodings.

Remark 4.2. We do not provide a definition of authenticity because it is a straightforward extension of our scheme and is already achieved by [GVW13]. We focus on circuit and input privacy, which have not been achieved by previous work.

Recall the class of circuits $\mathcal{C}_{n, d(n)}$ defined for Corollary 3.2.

Theorem 4.3. There exists a polynomial p , such that for every depth $d = d(n)$ function of the input size n , there is a reusable garbling scheme for any class of boolean circuits $\{\mathcal{C}_{n, d}\}_{n \in \mathbb{N}}$, assuming there is a fully secure single-key functional encryption scheme for any class of boolean circuits $\{\mathcal{C}_{n, p(d)}\}_{n \in \mathbb{N}}$.

Corollary 4.4 (The LWE Instantiation). For every integer $n \in \mathbb{N}$, polynomial function $d = d(n)$, there is a reusable garbling scheme for the class $\mathcal{C}_{n, d(n)}$, under the following assumption: there is a constant $0 < \epsilon < 1$ such that for every sufficiently large ℓ , the approximate shortest vector problem gapSVP in ℓ dimensions is hard to approximate to within a $2^{O(\ell^\epsilon)}$ factor in time $2^{O(\ell^\epsilon)}$ in the worst case.

The proof of this corollary follows from Theorem 4.3 when instantiating the functional encryption scheme with the one from Corollary 3.2.

Denote by *universal reusable garbling scheme*, a reusable garbling scheme for the class of all polynomial-sized circuits. Then, the following corollary follows directly from Theorem 4.3:

Corollary 4.5 (Universal reusable garbled circuits). *If there is a universal single-key fully secure functional encryption scheme, there is a universal reusable garbling scheme.*

Notice that our functional encryption tool (FE) already gives reusable garbled circuits with input privacy but no circuit privacy: the garbling of C is $\text{FE.KeyGen}(\text{fmsk}, C)$, whereas the encoding of the input x is $\text{FE.Enc}(\text{fmpk}, x)$. The fact that our scheme is single-key does not pose a limitation because the single-key corresponds to the circuit to garble (and any input encoding need only work with one garbled circuit). Since the single-key for one function works with an arbitrary number of encrypted inputs, the resulting garbled circuit is reusable.

However, the problem is that FE does not hide the circuit C , which is a required property of garbling schemes. The insight in achieving circuit privacy is to use the input-hiding property of the FE scheme to hide the circuit as well. The first idea that comes to mind is to hide C by including it in the ciphertext together with the input x . Specifically, instead of providing a key for circuit C , the encryptor runs FE.KeyGen on a universal circuit U that on input (C, x) computes $C(x)$. Notice that U can be public because it carries no information about C other than its size. Now the encryption of x consists of an encryption of (C, x) using FE.Enc . In this way, we can see that the resulting garbled circuit satisfies the correctness property. Moreover, for security, FE hides the input (C, x) so it would hide the circuit C as well.

Nevertheless, this approach is not useful because the encoding is as large as the circuit C (in particular, RGb.Enc no longer satisfies the efficiency property in Def. 2.6). Moreover, in this case, the standard one-time garbling schemes would be enough because one could produce a fresh garbled circuit with each ciphertext.

To overcome this problem, the idea is to provide, together with the ciphertext of x , the ability to decrypt C rather than the entire description of C . Specifically, let E be the encryption of the circuit C with a semantically secure symmetric encryption scheme under a secret key sk . The garbling of C consists of running the key generation FE.KeyGen on a circuit U_E that includes E and works as follows. On input (x, sk) the circuit U_E decrypts E to obtain C , and outputs the result of running C on x . Even though $\text{FE.KeyGen}(\text{fmsk}, U_E)$ does not hide U_E , the description of U_E does not leak C because C is encrypted. An encoding by RGb.Enc of x thus consists of running the encryption algorithm FE.Enc on (x, sk) .

4.1 Construction

We construct a reusable garbling scheme $\text{RGb} = (\text{RGb.Garble}, \text{RGb.Enc}, \text{RGb.Eval})$ as follows. Let $\text{E} = (\text{E.KeyGen}, \text{E.Enc}, \text{E.Dec})$ be a semantically secure symmetric-key encryption scheme.

Garbling $\text{RGb.Garble}(1^\kappa, C)$:

1. Generate FE keys $(\text{fmpk}, \text{fmsk}) \leftarrow \text{FE.Setup}(1^\kappa)$ and a secret key $\text{sk} \leftarrow \text{E.KeyGen}(1^\kappa)$.
2. Let $E := \text{E.Enc}(\text{sk}, C)$.
3. Define U_E to be the following universal circuit:

U_E takes as input a secret key sk and a value x :

- (a) Compute $C := \text{E.Dec}(\text{sk}, E)$.
- (b) Run C on x .

4. Let $\Gamma \leftarrow \text{FE.KeyGen}(\text{fmsk}, U_E)$ be the reusable garbled circuit.

5. Output $\text{gsk} := (\text{fmpk}, \text{sk})$ as the secret key and Γ as the garbling of C .

Encoding $\text{Rb.Enc}(\text{gsk}, x)$: Compute $c_x \leftarrow \text{FE.Enc}(\text{fmpk}, (\text{sk}, x))$ and output c_x .

Evaluation $\text{Rb.Eval}(\Gamma, c_x)$: Compute and output $\text{FE.Dec}(\Gamma, c_x)$.

The existence of a semantically secure encryption scheme does not introduce new assumptions because the FE scheme itself is a semantically secure encryption scheme if no key (computed by FE.KeyGen) is ever provided to an adversary.

Tightness of the scheme. The astute reader may have observed that the resulting scheme requires that the encodings be generated in the secret key setting because the encoding of x includes sk . It turns out that generating encodings privately is in fact necessary; if the encodings were publicly generated, the power of the adversary would be the same as in traditional obfuscation, which was shown impossible [BGI⁺01, GK05] (see discussion in Sec. 1.1.2).

One might wonder though, whether a reusable garbling scheme exists where the encoding generation is secret key, but Rb.Garble is public key. We prove in Sec. 4.3 that this is also not possible based on the impossibility result of [AGVW12]; hence, with regard to public versus private key, our reusable garbling result is tight.

4.2 Proof

Proof of Theorem 4.3. We first argue the scheme satisfies the correctness and efficiency properties in Def. 2.6.

Claim 4.6. *The above scheme Rb is a correct and efficient garbling scheme.*

Proof. We can easily see correctness of Rb.Eval :

$$\begin{aligned} \text{Rb.Eval}(\Gamma, c_x) &= \text{FE.Dec}(\Gamma, c_x) \quad (\text{by the definition of Rb.Eval}) \\ &= U_E(\text{sk}, x) \quad (\text{by the correctness of FE}) \\ &= C(x) \quad (\text{by the definition of } U_E). \end{aligned}$$

The efficiency of Rb depends on the efficiency of the FE.Enc algorithm and the length of gsk depends on the FE.Setup . If the runtime of FE.Enc does not depend on the class of circuits to be computed at all, the same holds for Rb.Enc 's efficiency. If FE.Enc and FE.Setup depend on the depth of the circuits to be computed, as is the case in our LWE instantiation, Rb.Enc 's runtime and $|\text{gsk}|$ also depend on the depth of the circuits, but still remain independent of the size of the circuits, which could potentially be much larger. \square

We can see that to obtain a Rb scheme for circuits of depth d , we need a FE scheme for polynomially deeper circuits: the overhead comes from the fact that U is universal and it also needs to perform decryption of E to obtain C .

To prove security, we need to construct a simulator $S = (S_1, S_2)$ satisfying Def. 4.1, assuming there is a simulator Sim_{FE} that satisfies Def. 2.13.

To produce a simulated garbled circuit $\tilde{\Gamma}$, S_1 on input $(1^\kappa, 1^{|C|})$ runs:

1. Generate fresh fmpk , fmsk , and sk as in Rb.Garble .
2. Compute $\tilde{E} := \text{E.Enc}(\text{sk}, 0^{|C|})$. (The reason for encrypting $0^{|C|}$ is that S_1 does not know C).
3. Compute and output $\tilde{\Gamma} \leftarrow \text{FE.KeyGen}(\text{fmsk}, U_{\tilde{E}})$.

S_2 receives queries for values $x_1, \dots, x_t \in \{0, 1\}^*$ for some t and needs to output a simulated encoding for each of these. To produce a simulated encoding for x_i , S_2 receives inputs $(C(x_i), 1^{|x_i|}$, and the latest simulator's state) and invokes the simulator Sim_{FE} of the FE scheme and outputs

$$\tilde{c}_x := \text{Sim}_{\text{FE}}(\text{fmpk}, \text{fsk}_{U_{\tilde{E}}}, U_{\tilde{E}}, C(x), 1^{|\text{sk}|+|x_i|}).$$

A potentially alarming aspect of this simulation is that S generates a key for the circuit $0^{|C|}$. Whatever circuit $0^{|C|}$ may represent, it may happen that there is no input x to $0^{|C|}$ that results in the value $C(x)$. The concern may then be that Sim_{FE} may not simulate correctly. However, this is not a problem because, by semantic security, E and \tilde{E} are computationally indistinguishable so Sim_{FE} must work correctly, otherwise it breaks semantic security of the encryption scheme E .

We now prove formally that the simulation satisfies Def. 4.1 for any adversary $A = (A_1, A_2)$. Let us assume that the α output of A_2 is its view, namely, all the information A_2 receives in the protocol, $(C, \text{state}_A, \Gamma, \{x_i, c_{x_i}\}_{i=1}^t)$. If the outcome of the real and ideal experiments are computationally indistinguishable in this case, then they are computationally indistinguishable for any other output strategy of A_2 because D can always run A_2 on its view since A_2 is p.p.t.. Therefore, we would like to show that:

$$\left\{ (C, \text{state}_A, \Gamma, \{x_i, c_{x_i}\}_{i=1}^t) \leftarrow \text{Exp}_{\text{RgB}, A}^{\text{real}}(1^\kappa) \right\}_\kappa \stackrel{c}{\approx} \left\{ (C, \text{state}_A, \tilde{\Gamma}, \{x_i, \tilde{c}_{x_i}\}_{i=1}^t) \leftarrow \text{Exp}_{\text{RgB}, A, S}^{\text{ideal}}(1^\kappa) \right\}_\kappa.$$

Game 0: The ideal game of Def. 4.1 with simulator S ; we recall that the output distribution in this case is

$$(C, \text{state}_A, \text{FE.KeyGen}(\text{fmsk}, U_{\tilde{E}}), \{x_i, \text{Sim}_{\text{FE}}(\text{fmpk}, \text{fsk}_{U_{\tilde{E}}}, U_{\tilde{E}}, C(x_i), 1^{|x_i|+|\text{sk}|})\}_{i=1}^t).$$

Game 1: The same as Game 0, but \tilde{E} is replaced with $E = E.\text{Enc}(\text{sk}, C)$. That is, the output distribution is

$$(C, \text{state}_A, \Gamma, \{x_i, \text{Sim}_{\text{FE}}(\text{fmpk}, \text{fsk}_{U_E}, U_E, C(x_i), 1^{|x_i|+|\text{sk}|})\}_{i=1}^t).$$

Game 2: The real game with our construction for RgB. It consists of the output distribution

$$(C, \text{state}_A, \Gamma, \{x_i, c_{x_i}\}_{i=1}^t).$$

First, let us argue that the distributions output by Game 0 and Game 1 are computationally indistinguishable. Note that these two distributions differ only in E and \tilde{E} . Since these distributions do not contain sk or any other function of sk other than E/\tilde{E} , by semantic security of the encryption scheme, we can show these two distributions are computationally indistinguishable. Finally, Lemma 4.7 proves that the outputs of Game 1 and Game 2 are also computationally indistinguishable, which concludes our proof.

Lemma 4.7. *Assuming FE is FULL-SIM-secure, the outputs of Game 1 and Game 2 are computationally indistinguishable.*

Proof. The proof of the lemma is by contradiction. We assume there exist p.p.t. adversaries $A = (A_1, A_2)$ and p.p.t. distinguisher D such that D with A can distinguish Game 1 and Game 2. Namely, there exists a polynomial $p(\cdot)$ such that, for infinitely many κ ,

$$|\Pr[D(\text{Exp}_{\text{FE}, A}^{\text{Game1}}(1^\kappa)) = 1] - \Pr[D(\text{Exp}_{\text{FE}, A}^{\text{Game2}}(1^\kappa)) = 1]| \geq 1/p(\kappa). \quad (5)$$

We construct adversaries that break the full security of the functional encryption scheme Def. 2.13. We call these adversaries $A^{\text{FE}} = (A_1^{\text{FE}}, A_2^{\text{FE}})$ and D^{FE} using the “FE” superscript to differentiate them from the adversaries distinguishing Game 1 and 2. In fact, we construct adversaries A^{FE} and D^{FE} that break a modified version of Def. 2.13: the modification is that A^{FE} can repeat Steps (4–5) as many times as it wishes and adaptively; more precisely, for the i -th repetition of Steps (4–5), A_2^{FE} can ask for an encryption of an input x_i where x_i could be determined based on the previous values and encryptions of x_1, \dots, x_{i-1} ; A_2^{FE} receives either a real encryption or a simulated encryption as in Step (5), but either all encryptions are real or all are simulated. We can see that if A^{FE} and D^{FE} break this modified definition, then they must break the original definition (with a polynomially smaller advantage): this implication follows from a standard hybrid argument possible because the encryption of x_i is public key.

On input fmpk , adversary A_1^{FE} works as follows:

1. Run A_1 on input 1^κ and obtain C and state_A .
2. Choose $\text{sk} \leftarrow \text{E.KeyGen}(1^\kappa)$, encrypt $E \leftarrow \text{E.Enc}(\text{sk}, C)$, and let U_E be the circuit described above.
3. Output function U_E and $\text{state}_A^{\text{FE}} := (\text{sk}, U_E, \text{state}_A)$.

On input $(\text{fsk}_{U_E}, \text{state}_A^{\text{FE}})$, adversary A_2^{FE} works as follows:

1. Let $\Gamma := \text{fsk}_{U_E}$.
2. Run A_2 on U_E, Γ and state_A by answering to its oracle queries as follows.
 - (a) Consider the i -th oracle query (x_i, state_A) . Output (x_i, sk) .
 - (b) Receive as input CT_i which is either the real ciphertext $c_i \leftarrow \text{FE.Enc}(\text{fmpk}, (x_i, \text{sk}))$ or the simulated ciphertext $\tilde{c}_i \leftarrow \text{Sim}_{\text{FE}}(\text{fmpk}, \Gamma, U_E, C(x_i), 1^{|x_i|+|\text{sk}|})$. Respond to A_2 with $(\text{CT}_i, \text{state}_A)$.
 - (c) Repeat these steps until A_2 finishes querying for encodings, and outputs α .
3. Output α .

Adversary D^{FE} is the same as D .

When the encodings CT_i are the ideal ciphertexts, we can see that $(A_1^{\text{FE}}, A_2^{\text{FE}})$ simulate perfectly Game 1; hence

$$\Pr[D^{\text{FE}}(\text{Exp}_{\text{FE}, A^{\text{FE}}}^{\text{ideal}}(1^\kappa)) = 1] = \Pr[D(\text{Exp}_{\text{FE}, A}^{\text{Game 1}}(1^\kappa)) = 1].$$

When the encodings CT_i are the real ciphertexts, $(A_1^{\text{FE}}, A_2^{\text{FE}})$ simulate perfectly Game 2 and thus

$$\Pr[D^{\text{FE}}(\text{Exp}_{\text{FE}, A^{\text{FE}}}^{\text{real}}(1^\kappa)) = 1] = \Pr[D(\text{Exp}_{\text{FE}, A}^{\text{Game 2}}(1^\kappa)) = 1].$$

By Eq. (5), we have

$$|\Pr[D^{\text{FE}}(\text{Exp}_{\text{FE}, A^{\text{FE}}}^{\text{ideal}}(1^\kappa)) = 1] - \Pr[D^{\text{FE}}(\text{Exp}_{\text{FE}, A^{\text{FE}}}^{\text{real}}(1^\kappa)) = 1]| \geq 1/p(\kappa),$$

which contradicts FULL-SIM-security of FE. \square

Having proved that Game 0 and Game 1 are computationally indistinguishable, and that Game 1 and Game 2 are computationally indistinguishable, we conclude that Game 0 and Game 2 are computationally indistinguishable, and therefore that garbling scheme Rgb is input- and circuit-private with reusability. \square

4.3 Impossibility of Public-Key Reusable Garbled Circuits

In this section, we show that a public-key reusable garbling scheme is impossible. Our argument is at a high level because it follows from existing results straightforwardly.

A public-key reusable garbling scheme would have the following syntax:

Definition 4.2 (Public-key garbling scheme). *A public-key garbling scheme PubGb for the class of circuits $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$, with \mathcal{C}_n a set of boolean circuits taking n bits as input, is a tuple of p.p.t. algorithms (PubGb.Setup, PubGb.Garble, PubGb.Enc, PubGb.Eval) such that*

- PubGb.Setup(1^κ): Takes as input the security parameter 1^κ and outputs a secret key gsk and a public key gpk.
- PubGb.Garble(gpk, C): Takes as input a public key gpk and a circuit C , and outputs the garbled circuit Γ of the circuit C .
- PubGb.Enc(gsk, x): Takes as input the secret key gsk and an input x , and outputs an encoding c_x .
- PubGb.Eval(Γ, c_x): Takes as input a garbled circuit Γ and an encoding c_x and outputs a value y .

Correctness. *For all polynomials $n(\cdot)$, for all sufficiently large security parameters κ , for $n = n(\kappa)$, for all circuits $C \in \mathcal{C}_n$, and for all $x \in \{0, 1\}^n$,*

$$\Pr[(\text{gsk}, \text{gpk}) \leftarrow \text{PubGb.Setup}(1^\kappa); \Gamma \leftarrow \text{PubGb.Garble}(\text{gpk}, C); c_x \leftarrow \text{PubGb.Enc}(\text{gsk}, x) : \text{PubGb.Eval}(\Gamma, c_x) = C(x)] = 1 - \text{negl}(\kappa).$$

The natural security definition of circuit-private definition of this new scheme is similar in flavor to Def. 2.13, but we do not elaborate. (In fact, this definition can be relaxed to not require input privacy for the impossibility result to still hold.)

The first step in the impossibility argument is to note that the syntax and correctness of a public-key garbling scheme is the same as the syntax of a functional encryption scheme (Def. 2.12) with the following correspondence of algorithms: PubGb.Setup corresponds to FE.Setup, PubGb.Garble corresponds to the encryption algorithm FE.Enc, PubGb.Enc corresponds to FE.KeyGen and PubGb.Eval corresponds to FE.Dec. Note that PubGb.Enc does not correspond to FE.Enc but to FE.KeyGen because PubGb.Enc is a secret key algorithm and FE.Enc is a public-key algorithm. Therefore, an encoding of an input x in the reusable garbling scheme corresponds to a secret key for a function f_x in the functional encryption scheme.

Moreover, considering this mapping, it is straightforward to show that a circuit-private public-key garbling scheme implies a secure functional encryption scheme. Since the reusable garbling scheme allows an arbitrary number of inputs being encoded, it implies that the functional encryption scheme can generate an arbitrary number of secret function keys sk_{f_x} ; furthermore, in this functional encryption scheme, the size of the ciphertexts does not depend on the number of keys generated (because this number is nowhere provided as input in the syntax of the scheme). This conclusion directly contradicts the recent impossibility result of Agrawal et al. [AGVW12]: they show that any functional encryption scheme that can securely provide q keys must have the size of the ciphertexts grow in q ; therefore, a reusable circuit-private public-key garbling scheme is unachievable.

5 Token-Based Obfuscation

Following the discussion of obfuscation in Sec. 1.1.2, the purpose of this section is to cast reusable garbled circuits in the form of obfuscation and to show that this provides a new model for obfuscation, namely *token-based obfuscation*.

Reusable garbled circuits come close to obfuscation: a reusable garbled circuit hides the circuit while permitting circuit evaluation on an arbitrary number of inputs. While they come close, reusable garbled circuits do not provide obfuscation, because the encoding of each input requires knowledge of the secret key: namely, to run an obfuscated program on an input, one needs to obtain a token for the input from the obfuscator. This requirement of our scheme is in fact necessary: as argued in the tightness discussion in Sec. 4, a scheme in which one can publicly encode inputs is impossible because it falls directly onto known impossibility results for obfuscation.

Therefore, we propose a new token-based model for obfuscation. The idea is for a program vendor to obfuscate his program and provide tokens representing rights to run this program on specific inputs. For example, consider the case when some researchers want to compute statistics on a database with sensitive information. The program to be obfuscated consists of the database service program with the secret database hardcoded in it, U_{DB} . When researchers want to compute statistics x , they request a token for x from the database owner. Using the obfuscated program and the token, the researchers can compute $U_{DB}(x)$, the statistics result by themselves without having to contact the owner again. It is crucial that the time to compute the token for x is much smaller than the time to compute U_{DB} on x , so that the owner does not have to do a lot of work. We also note, that in certain cases, one has to anyways request such a token from the owner for other reasons: for example, the database owner can check that the statistics the researchers want to compute is not too revealing and grant a token only if this is the case.

Let us compare the token-based obfuscation model with the obfuscation model resulting from using FHE. With FHE, the obfuscation of a program is the FHE encryption of the program. When the client wants to feed an input to the obfuscated program, the client can encrypt this input by herself using the FHE public-key and does not need to obtain a token from the obfuscator. To run the program, the client performs FHE evaluation of a universal circuit on the encrypted program and the encrypted input, thus obtaining an encrypted result. The client cannot decrypt the result by herself and thus needs to contact the obfuscator for this decryption – this process consists of two messages. In our token-based model, if the obfuscator knows a priori the inputs for which to send tokens to the client (e.g., when distributing permissions for certain computations), the whole protocol consists of one message only because the client can compute and decrypt the result by herself. Another difference between these two obfuscation models is that, in the token-based model, the obfuscator needs to be available only at the beginning of the computation (when giving out tokens), whereas in the FHE model, the obfuscator has to be online at the end of the computation to decrypt the result.

5.1 Definition

We now provide the definition for token-based obfuscation and the desired simulation security. These definitions are very similar to the definitions for reusable garbled circuits (Def. 2.6 and Def. 4.1): the syntax, correctness and efficiency are the same except that garbling schemes have an additional Eval algorithm.

Definition 5.1 (Token-based Obfuscation). *A token-based obfuscation scheme for the class of circuits $\{\mathcal{C}_n\}_{n \in \mathbb{N}}$ with $\mathcal{C}_n : \{0, 1\}^n \rightarrow \{0, 1\}$ is a pair of p.p.t. algorithms (tOB.Obfuscate, tOB.Token) such that*

- $\text{tOB.Obfuscate}(1^\kappa, C)$: Takes as input the security parameter 1^κ , and a circuit $C \in \mathcal{C}_n$, and outputs a secret key osk and the obfuscation O of the circuit C .

- $\text{tOB.Token}(\text{osk}, x)$: Takes as input the secret key osk and some input $x \in \{0, 1\}^n$, and outputs tk_x .

Efficiency. The running time of tOB.Token is independent of the size of C .

Correctness. For all polynomials $n(\cdot)$, for all sufficiently large security parameters κ , if $n = n(\kappa)$, for all circuits $C \in \mathcal{C}_n$, and for all $x \in \{0, 1\}^n$,

$$\Pr[(\text{osk}, O) \leftarrow \text{tOB.Obfuscate}(1^\kappa, C); \text{tk}_x \leftarrow \text{tOB.Token}(\text{osk}, x) : O(\text{tk}_x) = C(x)] = 1 - \text{negl}(\kappa).$$

Remark 5.1. We could use an alternative definition of token-based obfuscation that separates the generation of osk (in an additional tOB.Setup algorithm with input the security parameter) from the tOB.Obfuscate algorithm. Such a formulation would force osk and thus the token computation $\text{tOB.Token}(\text{osk}, x)$ to be independent of the circuit obfuscated; moreover, C could be chosen later, even after all inputs x have been encrypted with tOB.Token .

Our construction satisfies this definition as well because it generates the secret key osk independent of C .

However, we did not choose such a formulation because we wanted to be consistent with the definition of obfuscation, which does not have a separate setup phase.

Intuitively, in a secure token-based obfuscation scheme, an adversary does not learn anything about the circuit C other than $C(x)$ and the size of C .

Definition 5.2 (Secure token-based obfuscation). Let tOB be a token-based obfuscation scheme for a family of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n \in \mathbb{N}}$. For $A = (A_1, A_2)$ and $S = (S_1, S_2)$, pairs of p.p.t. algorithms, consider the following two experiments:

$\text{Exp}_{\text{tOB}, A}^{\text{real}}(1^\kappa)$:	$\text{Exp}_{\text{tOB}, A, S}^{\text{ideal}}(1^\kappa)$:
1: $(C, \text{state}_A) \leftarrow A_1(1^\kappa)$	1: $(C, \text{state}_A) \leftarrow A_1(1^\kappa)$
2: $(\text{osk}, O) \leftarrow \text{tOB.Obfuscate}(1^\kappa, C)$	2: $(\tilde{O}, \text{state}_S) \leftarrow S_1(1^\kappa, 1^{ C })$
3: $\alpha \leftarrow A_2^{\text{tOB.Token}(\text{osk}, \cdot)}(C, O, \text{state}_A)$	3: $\alpha \leftarrow A_2^{\text{OS}(\cdot, C)[[\text{state}_S]]}(C, \tilde{O}, \text{state}_A)$
4: Output α	4: Output α

In the above, $\text{OS}(\cdot, C)[[\text{state}_S]]$ is an oracle that on input x from A_2 , runs S_2 with inputs $C(x)$, $1^{|x|}$, and the current state of S , state_S . S_2 responds with tk_x and a new state state'_S which OS will feed to S_2 on the next call. OS returns tk_x to A_2 .

We say that the token-based obfuscation tOB is secure if there exists a pair of p.p.t. simulators $S = (S_1, S_2)$ such that for all pairs of p.p.t. adversaries $A = (A_1, A_2)$, the following two distributions are computationally indistinguishable:

$$\left\{ \text{Exp}_{\text{tOB}, A}^{\text{real}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx} \left\{ \text{Exp}_{\text{tOB}, A, S}^{\text{ideal}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}}.$$

Note that, in this security definition, a token tk_x hides x as well because S_2 never receives x . This is usually not required of obfuscation, but we achieve this property for free.

5.2 Scheme

The construction of a token-based obfuscation scheme is very similar to the construction of reusable garbled circuits, the technical difference being minor: we need to specify how to construct the algorithm tOB.Obfuscate from Rb.Garble and Rb.Eval . We construct a token-based obfuscation $\text{tOB} =$

(tOB.Obfuscate, tOB.Token) as follows based on a reusable garbled scheme $\text{RGb} = (\text{RGb.Garble}, \text{RGb.Enc}, \text{RGb.Eval})$. The token algorithm tOB.Token is the same as RGb.Enc.

Obfuscation tOB.Obfuscate($1^\kappa, C \in \mathcal{C}_n$):

1. Let $(\Gamma, \text{sk}) \leftarrow \text{RGb.Garble}(1^\kappa, C)$.
2. Construct the circuit O (the obfuscation of C) as follows. The circuit O has Γ hardcoded. It takes as input a token tk_x , computes $\text{RGb.Eval}(\Gamma, \text{tk}_x)$, and outputs the result.
3. Output sk as the secret key, and the description of O as the obfuscation of C .

Since the construction is essentially the same as the one of reusable garbled circuits and the security is the same, the same claims and proofs as for reusable garbled circuits hold here, based on Theorem 4.3 and Corollary 4.4. We state them here for completeness.

Claim 5.2. *Assuming a reusable garbling scheme for the class of circuits \mathcal{C} , there is a token-based obfuscation scheme for \mathcal{C} .*

Recall the class of circuits $\mathcal{C}_{n,d(n)}$ defined for Corollary 3.2.

Corollary 5.3 (The LWE Instantiation). *For every integer $n \in \mathbb{N}$, polynomial function $d = d(n)$, there is a token-based obfuscation scheme for the class $\mathcal{C}_{n,d(n)}$, under the following assumption: there is a constant $0 < \epsilon < 1$ such that for every sufficiently large ℓ , the approximate shortest vector problem gapSVP in ℓ dimensions is hard to approximate to within a $2^{O(\ell^\epsilon)}$ factor in time $2^{O(\ell^\epsilon)}$ in the worst case.*

Denote by *universal token-based obfuscation scheme*, a token-based obfuscation scheme for the class of all polynomial-sized circuits. Then,

Corollary 5.4 (Universal token-based obfuscation). *If there is a universal fully secure single-key functional encryption scheme, there is a universal token-based obfuscation scheme.*

6 Computing on Encrypted Data in Input-Specific Time

We initiate the study of fully homomorphic encryption where the runtime of the homomorphic evaluation is input-specific rather than worst-case time. We show how to use our functional encryption scheme to evaluate Turing machines on encrypted data in input-specific time.

Let us recall the setting of computation on encrypted data. A client gives various encrypted inputs and a function f to an evaluator. The evaluator should compute f on the encrypted inputs and return the encrypted result, while learning nothing about the inputs.

Fully homomorphic encryption has been the main tool used in this setting. It was first constructed in a breakthrough work by Gentry [Gen09] and refined in subsequent work [DGHV10, SS10b, BV11a, Vai11, BGV12, GHS12a, GHS12b]. Since then, FHE has found many great applications to various problems.

However, one of the main drawbacks of FHE is that when evaluating a Turing machine (TM) over encrypted data, the running time is at least the worst-case running time of the Turing machine over all inputs. The reason is that, one needs to transform the TM into a circuit. If t_{\max} is the maximum running time of the TM on inputs of a certain size—namely, the running time on the worst-case input—then the size of the resulting circuit is at least t_{\max} . Thus, even if the TM runs in a short time on most of the inputs, but for a very long time (t_{\max}) on only one input, *homomorphic evaluation will still run in t_{\max} for all inputs*. This property

often results in inefficiency in practice; for example, consider a TM having a loop that depends on the input. For specific inputs, it can loop for a very long time, but for most inputs it does not loop at all.

As a result, researchers have tried to find input-specific schemes. A first observation is that this goal is impossible: input-specific evaluation implies that the evaluator learns the runtime of the TM on each input, which violates CPA-security of the homomorphic scheme (Def. 2.5). Hence, we must relax the security definition and allow the evaluator to learn the runtime for each input, but require that *the evaluator learns nothing else besides the running time*. This goal is not possible with FHE because the evaluator cannot decrypt any bit of information, so it cannot tell whether the computation finished or not; thus, we must look for new solutions.

A second observation is that the evaluator must no longer be able to evaluate TMs of his choice on the client's data: if he could, the evaluator would run TMs whose running times convey the value of the input x (for example, the evaluator could run $|x|$ TMs, where the i -th TM stops early if the i -th bit of x is zero, and otherwise, it stops later; in this way, the evaluator learns the exact value of x).

Based on these observations, we can see that functional encryption is the natural solution: it hides the inputs to the computation, enables the evaluator to decrypt the running time, and requires the evaluator to obtain a secret key from the client to evaluate each TM.

Due to the impossibility result for functional encryption [AGVW12] discussed in Sec. 1, the client cannot give keys for an arbitrary number of Turing machines to the evaluator. The best we can hope to achieve is for the client to provide a single key for a function to the evaluator (or equivalently, for a constant number q of keys if the client runs the scheme q times). Fortunately, the single-key restriction does not mean that the client can evaluate only one Turing machine. In fact, the client can give a key to the evaluator for a universal Turing machine U that takes as input a TM M and a value x , and outputs $M(x)$. Then, the client must specify together with each input x the TM M he wants to run on x . Such a strategy is even desirable in certain cases: the client may not want the evaluator to compute a TM on every input the client has provided and learn the running time on that input; the client may prefer to specify what inputs to run each Turing machine on.

Using our functional encryption scheme, we achieve a construction that enables computation in input-specific time. We call such a scheme *Turing machine homomorphic encryption*, or shortly TMFHE.

As discussed (Corollary 3.2), our functional encryption scheme is succinct in that the ciphertexts grow with the depth of the circuit rather than the size of the circuit. Therefore, our input-specific computation is useful only for Turing machines that can be represented in circuits whose depths are smaller than the running time – because otherwise the client would have to do a lot of work and could instead just run the Turing machine on its own. Moreover, for these machines, we cannot use the Pippenger-Fischer [PF79] transformation because the resulting circuits have depth roughly equal to the running time of the transformed machines. Specifically, our input-specific scheme makes sense for the following class of circuits, with a bound on their depth.

Definition 6.1 (*d*-depth-bounded class of Turing machines). *A finite class of Turing machines \mathcal{M} is d -depth-bounded for a function d , if there exists a class of efficiently computable transformations $\{\mathcal{T}_n\}_{n \in \mathbb{N}}$ with $\mathcal{T}_n : \mathbb{N} \rightarrow \{\text{all circuits}\}$ such that $\mathcal{T}_n(t) = C_{n,t}$ where $C_{n,t}$ is a circuit as follows.*

- *On input a Turing machine $M \in \mathcal{M}$ and a value $x \in \{0, 1\}^n$, $C_{n,t}$ outputs $M(x)$ if M on input x stops in t steps, or \perp otherwise.*
- *The depth of $C_{n,t}$ is at most $d(n)$ and the size of $C_{n,t}$ is $\tilde{O}(t)$.*

Remark 6.1. *Notice that, if we remove the depth constraint (but still keep the circuit size constraint), any finite class of Turing machines satisfies the definition because of the Pippenger-Fischer transformation*

applied to the universal circuit of this class of Turing machines. Specifically, let U_t be a universal Turing machine that runs any given machine $M \in \mathcal{M}$ for t steps. This machine has $O(t)$ running time and when applying the Pippenger-Fischer transformation [PF79] to it, we get a circuit of size $O(t \log t)$.

We next present our construction. For completeness, we provide formal definitions and proofs of our theorems and claims in Appendix C. Our security notion (Def. C.2 in the appendix) is called *runtime-CPA security*, which straightforwardly captures the fact that the evaluator should learn nothing about the computation besides the running time.

6.1 Construction

A TMFHE scheme consists of four algorithms: $\text{TMFHE} = (\text{TMFHE.KeyGen}, \text{TMFHE.Enc}, \text{TMFHE.Eval}, \text{TMFHE.Dec})$. The client runs TMFHE.KeyGen once in an offline preprocessing stage. Later, in the online phase, the client sends a potentially large number of encrypted inputs to the evaluator. For every input (x, M) consisting of a value x and a Turing machine M , the client runs TMFHE.Enc to encrypt the input and then TMFHE.Dec to decrypt the result from the evaluator. The evaluator runs TMFHE.Eval to evaluate M on x homomorphically in input-specific running time. The work of the client in the offline phase is proportional to t_{\max} , the worst-case input running time. However, for each input in the online phase, the client does little work (independent of the running time of M) and thus the cost is amortized.

We first provide intuition for our construction. As mentioned, we use our functional encryption scheme FE to enable the evaluator to determine at various intermediary points whether the computation finished or not. For each intermediary step, the client has to provide the evaluator with a function secret key fsk (using the FE scheme) for a function that returns a bit indicating whether the computation has finished. However, if the client provides a key for every computation step, the offline work of the client becomes quadratic in t_{\max} , which can be very large in certain cases. The idea is to choose intermediary points spaced at exponentially increasing intervals. In this way, the client generates only a logarithmic number of keys, while the evaluator runs in roughly twice the time of M on an input.

As part of TMFHE.Enc , besides providing the FE encryptions for a pair (M, x) , the client also provides a homomorphic encryption for x and the machine M , so that once the evaluator learns the running time of M on x , it can then perform the homomorphic computation on x in that running time.

We present our construction for a class of d -depth-bounded Turing machines. By Def. 6.1, such a class has a transformation \mathcal{T}_n that enables transforming a universal TM into a circuit. Let FHE be any homomorphic encryption scheme (as defined in Sec. 2.3) for circuits of depth d and let FE be any functional encryption scheme for circuits of depth d . For simplicity, we present our scheme for Turing machines that output only one bit; we discuss in Sec. 6.3 multiple output bits and how to avoid having the output size be worst case.

Key generation $\text{TMFHE.KeyGen}(1^\kappa, 1^n, 1^{t_{\max}})$ takes as input the security parameter κ , an input size n , and a maximum time bound t_{\max} .

1. Let $\tau = \lceil \log t_{\max} \rceil$. For each $i \in [\tau]$, let $D_i = \mathcal{T}_n(2^i)$ be the circuit that outputs $M(x)$ if M finishes in 2^i steps on input x or \perp otherwise. Construct circuit C_i based on D_i : the circuit C_i , on input a TM M and a value x , outputs 1 if M finished in 2^i steps when running on input x or 0 otherwise; C_i is the same as circuit D_i but it just outputs whether the first output bit of C_i is non- \perp or \perp , respectively.
2. Generate functional encryption secret keys for C_1, \dots, C_τ by running:

$$(\text{fmpk}_i, \text{fmsk}_i) \leftarrow \text{FE.Setup}(1^\kappa) \text{ and } \text{fsk}_i \leftarrow \text{FE.KeyGen}(\text{fmsk}_i, C_i) \text{ for } i \in [\tau].$$

3. Generate FHE keys $(\text{hsk}, \text{hpk}) \leftarrow \text{FHE.KeyGen}(1^\kappa)$.
4. Output the tuple $\text{PK} := (\text{fmpk}_1, \dots, \text{fmpk}_\tau, \text{hpk})$ as the public key, $\text{EVK} := (\text{fsk}_1, \dots, \text{fsk}_\tau, \text{hpk})$ as the evaluation key, and $\text{SK} := \text{hsk}$ as the secret key.

Encryption $\text{TMFHE.Enc}(\text{PK}, M, x)$: takes as input the public key PK of the form $(\{\text{fmpk}_i\}_i, \text{hpk})$, a TM M and a value x of n bits long.

1. Let $\hat{x} \leftarrow (\text{FHE.Enc}(\text{hpk}, x_1), \dots, \text{FHE.Enc}(\text{hpk}, x_n))$, where x_i is the i -th bit of x . Similarly, let $\hat{M} \leftarrow (\text{FHE.Enc}(\text{hpk}, M_1), \dots, \text{FHE.Enc}(\text{hpk}, M_n))$, which is the homomorphic encryption of M (the string description of TM M) bit by bit.
2. Compute $c_i \leftarrow \text{FE.Enc}(\text{fmpk}_i, (M, x))$ for $i \in [\tau]$.
3. Output the ciphertext $c = (\text{"enc"}, \hat{x}, \hat{M}, c_1, \dots, c_\tau)$.

Evaluation $\text{TMFHE.Eval}(\text{EVK}, c)$: takes as input an evaluation key EVK of the form $(\{\text{fsk}_i\}_i, \text{hpk})$ and a ciphertext c of the form $(\text{"enc"}, \hat{x}, \hat{M}, c_1, \dots, c_\tau)$.

1. Start with $i = 1$. Repeat the following:
 - (a) $b \leftarrow \text{FE.Dec}(\text{fsk}_i, c_i)$.
 - (b) If $b = 1$, (computation finished and we can now evaluate homomorphically on \hat{x})
 - i. Compute D_i , the circuit that evaluates a Turing machine in \mathcal{M} for 2^i steps, using $\mathcal{T}_n(2^i)$.
 - ii. Evaluate and output $(\text{"eval"}, \text{FHE.Eval}(\text{hpk}, D_i, (\hat{M}, \hat{x})))$.
 - (c) Else ($b = 0$), proceed to the next i .

Decryption $\text{TMFHE.Dec}(\text{SK}, c)$: takes as input a secret key $\text{SK} = \text{hsk}$ and a ciphertext c of the form $(\text{"enc"}, \hat{x}, \hat{M}, c_1, \dots, c_\tau)$ or $(\text{"eval"}, c)$.

1. If the ciphertext is of type "enc", compute and output $\text{FHE.Dec}(\text{hsk}, \hat{x})$.
2. Else (the ciphertext is of type "eval"), compute and output $\text{FHE.Dec}(\text{hsk}, c)$.

6.2 Results

We now state our results.

Theorem 6.2. *For any class of d -depth-bounded Turing machines that take n bits of input and produce one bit of output, there is a Turing machine homomorphic encryption scheme, assuming the existence of a fully secure functional encryption scheme FE for any class of circuits of depth d , and an d -leveled homomorphic encryption scheme FHE, where:*

- The online work of the client is

$$(n + \log t_{\max}) \cdot \text{poly}(\kappa, d(n))$$

- The online work of the server in evaluating M on an encryption of x is

$$\text{poly}(n, d(n), \text{time}(M, x)),$$

where $\text{time}(M, x)$ is the runtime of M on x .

This theorem shows that our TMFHE scheme comes as a reduction from any functional encryption scheme. The proof of this theorem is in Appendix C. We can see that the work of the client is indeed smaller than computing the circuit especially if the polynomial d is smaller than the running time. Moreover, we can also see that the server runs in input-specific time: the evaluation time depends on the actual running time and the depth of the circuit.

When instantiating our TMFHE construction with our functional encryption FE construction from Sec. 3 and using Corollary 3.2, we obtain a scheme under an LWE assumption.

Corollary 6.3 (LWE Instantiation). *For every integer $n \in \mathbb{N}$ and polynomial function $d = d(n)$, there is a Turing machine homomorphic encryption scheme for any class of d -depth-bounded Turing machines, under the following assumption: there is a constant $0 < \epsilon < 1$ such that for every sufficiently large ℓ , the approximate shortest vector problem gapSVP in ℓ dimensions is hard to approximate to within a $2^{O(\ell^\epsilon)}$ factor in time $2^{O(\ell^\epsilon)}$ in the worst case.*

Remark 6.4. *If the underlying FE scheme is selectively secure (Def. 2.14), one can still obtain an input-specific homomorphic encryption scheme, but with selective security; namely, the scheme would achieve a modified version of Def. C.2 in Appendix C (the adversary A must choose x before seeing EVK and PK). The scheme would then be secure under the following assumption: there is a constant $0 < \epsilon < 1$ such that for every sufficiently large ℓ , the approximate shortest vector problem gapSVP in ℓ dimensions is hard to approximate to within a $2^{O(\ell^\epsilon)}$ factor in the worst case by polynomial-time adversaries.*

Let us discuss what kind of Turing machines classes are d -depth-bounded.

Fact 6.5. *The class of Turing machines running in log-space is \log^2 -depth-bounded.*

This fact follows directly from the known relation that the LOGSPACE complexity class is in NC2.

In general, the following pattern of computation would fit in d -depth-boundedness and would benefit from input-specific evaluation. Consider a computation that on different types of inputs, it performs different kinds of computation; all these computations are of the same (shallow) depth, but the computation can be much larger in one case.

A few remarks are in order:

Remark 6.6. *Denote by universal TMFHE scheme to be a scheme for any finite class of Turing machines. Based on Remark 6.1, we can see that if there is a universal succinct functional encryption scheme and a fully homomorphic scheme, there is a universal TMFHE scheme with online client and server work independent of depth:*

- The online work of the client becomes

$$(n + \log t_{\max}) \cdot \text{poly}(\kappa)$$

- The online work of the server in evaluating M on an encryption of x becomes

$$\text{poly}(n, \text{time}(M, x)),$$

where $\text{time}(M, x)$ is the runtime of M on x .

6.3 Input-Dependent Output Size

The construction above considered Turing machines that output only one bit. To allow TMs that output more than one bit, one can simply use the standard procedure of running one instance of the protocol for each bit of the output. However, as with running time, this would result in repeating the protocol as many times as the worst-case output size for every input. Certain inputs can result in small outputs while others can result in large outputs, so it is desirable to evaluate in *input-specific output size*.

We can use the same approach as above to obtain input-specific output size: The client provides keys to the evaluator to decrypt the size of the output. Then, the evaluator can simply use homomorphic evaluation on a circuit whose output size is the determined one.

Acknowledgments

This work was supported by an NSERC Discovery Grant, by DARPA awards FA8750-11-2-0225 and N66001-10-2-4089, by NSF awards CNS-1053143 and IIS-1065219, and by Google.

References

- [AFV11] Shweta Agrawal, David Mandell Freeman, and Vinod Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, pages 21–40, 2011.
- [AGVW12] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. *Cryptology ePrint Archive, Report 2012/468*, 2012.
- [AKS01] Miklós Ajtai, Ravi Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *STOC*, pages 601–610, 2001.
- [Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *FOCS*, pages 298–307, 2003.
- [BCG⁺11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *ASIACRYPT*, pages 722–739, 2011.
- [BFL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *CRYPTO*, pages 278–291, 1993.
- [BG⁺01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.
- [BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *ITCS*, pages 309–325, 2012.
- [BHH10] Boaz Barak, Iftach Haitner, Dennis Hofheinz, and Yuval Ishai. Bounded key-dependent message security. In *EUROCRYPT*, pages 423–444, 2010.
- [BHR12] Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Garbling schemes. *Cryptology ePrint Archive, Report 2012/265*, 2012.
- [BKW03] Avrim Blum, Adam Kalai, and Hal Wasserman. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM*, 50(4):506–519, 2003.
- [Bra12] Zvika Brakerski. Fully homomorphic encryption without modulus switching from classical GapSVP. In *CRYPTO*, pages 868–886, 2012.
- [BSW] Dan Boneh, Amit Sahai, and Brent Waters.

- [BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the 28th IEEE Symposium on Security and Privacy*, pages 321–334, 2007.
- [BV11a] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS*, pages 97–106, 2011.
- [BV11b] Zvika Brakerski and Vinod Vaikuntanathan. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *CRYPTO*, pages 505–524, 2011.
- [BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.
- [CKVW10] Ran Canetti, Yael Tauman Kalai, Mayank Varia, and Daniel Wichs. On symmetric encryption and point obfuscation. In *TCC*, pages 52–71, 2010.
- [Dav12] Michael A. Davis. Cloud security: Verify, don’t trust. *Information Week*, August 2012. <http://reports.informationweek.com/abstract/5/8978/>.
- [DGHV10] Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully homomorphic encryption over the integers. In *EUROCRYPT*, pages 24–43, 2010.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [GGH12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. Cryptology ePrint Archive, Report 2012/610, 2012.
- [GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.
- [GHS12a] Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully homomorphic encryption with polylog overhead. In *EUROCRYPT*, 2012.
- [GHS12b] Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic evaluation of the AES circuit. Cryptology ePrint Archive, Report 2012/099, 2012.
- [GHV10] Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. A simple BGN-type cryptosystem from LWE. In *EUROCRYPT*, pages 506–522, 2010.
- [GIS⁺10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.
- [GJPS08] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *ICALP*, pages 579–591, 2008.
- [GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.
- [GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO*, pages 39–56, 2008.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *STOC*, pages 218–229, 1987.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, pages 89–98, 2006.
- [GR07] Shafi Goldwasser and Guy N. Rothblum. On best-possible obfuscation. In *TCC*, pages 194–213, 2007.
- [GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, August 2012.
- [GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.

- [KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.
- [LOS⁺10] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.
- [LP09] Yehuda Lindell and Benny Pinkas. A proof of security of Yao’s protocol for two-party computation. *J. Cryptol.*, 22:161–188, April 2009.
- [LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *STOC*, pages 1219–1234, 2012.
- [LW12] Allison B. Lewko and Brent Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, 2012.
- [MV10] Daniele Micciancio and Panagiotis Voulgaris. A deterministic single exponential time algorithm for most lattice problems based on Voronoi cell computations. In *STOC*, pages 351–358, 2010.
- [O’N10] Adam O’Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010.
- [OT09] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, pages 214–231, 2009.
- [OT10] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem. In *STOC*, pages 333–342, 2009.
- [PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *J. ACM*, 26(2):361–381, 1979.
- [Pri12] Privacy Rights Clearinghouse. Chronology of data breaches, 2012. <http://www.privacyrights.org/data-breach>.
- [PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC*, pages 422–439, 2012.
- [RAD78] R. Rivest, L. Adleman, and M. Dertouzos. On data banks and privacy homomorphisms. In *Foundations of Secure Computation*, pages 169–177. Academic Press, 1978.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *STOC*, pages 84–93, 2005.
- [SS10a] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM CCS*, pages 463–472, 2010.
- [SS10b] Damien Stehlé and Ron Steinfeld. Faster fully homomorphic encryption. In *ASIACRYPT*, pages 377–394, 2010.
- [SSW09] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC*, pages 457–473, 2009.
- [SV11] Nigel P. Smart and Frederik Vercauteren. Fully homomorphic SIMD operations. Cryptology ePrint Archive, Report 2011/133, 2011.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
- [SW12] Amit Sahai and Brent Waters. Attribute-based encryption for circuits from multilinear maps. Cryptology ePrint Archive, Report 2012/592, 2012.

- [Vai11] Vinod Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In *FOCS*, pages 5–16, 2011.
- [Ver] Verizon RISK Team. 2012 data breach investigations report. http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2012_en_xg.pdf.
- [Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In *PKC*, pages 53–70, 2011.
- [Wat12] Brent Waters. Functional encryption for regular languages. In *CRYPTO*, pages 218–235, 2012.
- [Wee05] Hoeteck Wee. On obfuscating point functions. In *STOC*, pages 523–532, 2005.
- [Yao82] Andrew C. Yao. Protocols for secure computations. In *FOCS*, pages 160–164, 1982.
- [Yao86] Andrew C. Yao. How to generate and exchange secrets (extended abstract). In *FOCS*, pages 162–167, 1986.

A Detailed Background on Learning With Errors (LWE)

The LWE problem was introduced by Regev [Reg05] as a generalization of “learning parity with noise” [BFKL93, BKW03, Ale03]. Regev showed that solving the LWE problem *on the average* is as hard as (quantumly) solving several standard lattice problems *in the worst case*. This result bolstered our confidence in the LWE assumption and generated a large body of work building cryptographic schemes under the assumption, culminating in the construction of a fully homomorphic encryption scheme [BV11a].

For positive integers ℓ and $q \geq 2$, a vector $\mathbf{s} \in \mathbb{Z}_q^\ell$, and a probability distribution χ on \mathbb{Z}_q , let $A_{\mathbf{s},\chi}$ be the distribution obtained by choosing a vector $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^\ell$ uniformly at random and a noise term $e \xleftarrow{\$} \chi$, and outputting $(\mathbf{a}, \langle \mathbf{a}, \mathbf{s} \rangle + e) \in \mathbb{Z}_q^\ell \times \mathbb{Z}_q$. A formal definition follows.

Definition A.1 (LWE). *For an integer $q = q(\ell)$ and an error distribution $\chi = \chi(\ell)$ over \mathbb{Z}_q , the learning with errors problem $\text{LWE}_{\ell,m,q,\chi}$ is defined as follows: Given m independent samples from $A_{\mathbf{s},\chi}$ (for some $\mathbf{s} \in \mathbb{Z}_q^\ell$), output \mathbf{s} with noticeable probability.*

The (average-case) decision variant of the LWE problem, denoted $\text{dLWE}_{\ell,m,q,\chi}$, is to distinguish (with non-negligible advantage) m samples chosen according to $A_{\mathbf{s},\chi}$ (for uniformly random $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^\ell$), from m samples chosen according to the uniform distribution over $\mathbb{Z}_q^\ell \times \mathbb{Z}_q$.

We denote by $\text{LWE}_{\ell,q,\chi}$ (resp. $\text{dLWE}_{\ell,q,\chi}$) the variant where the adversary gets oracle access to $A_{\mathbf{s},\chi}$, and is not a priori bounded in the number of samples.

For cryptographic applications we are primarily interested in the average case decision problem dLWE , where $\mathbf{s} \xleftarrow{\$} \mathbb{Z}_q^\ell$. We will also be interested in assumptions of the form: no t -time adversary can solve dLWE with non-negligible advantage, which we will call the t -hardness of dLWE .

There are known quantum [Reg05] and classical [Pei09] reductions between $\text{dLWE}_{\ell,m,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be (discretized versions of) the Gaussian distribution, which is B -bounded for an appropriate B . Since the exact distribution χ does not matter for our results, we state a corollary of the results of [Reg05, Pei09] in terms of the bound on the distribution.

Let $B = B(\ell) \in \mathbb{N}$. A family of distributions $\chi = \{\chi_\ell\}_{\ell \in \mathbb{N}}$ is called B -bounded if the support of χ_ℓ is (a subset of) $[-B(\ell), \dots, B(\ell)]$. Then:

Lemma A.1 ([Reg05, Pei09]). Let $q = q(\ell) \in \mathbb{N}$ be a product of co-prime numbers $q = \prod q_i$ such that for all i , $q_i = \text{poly}(\ell)$, and let $B \geq \ell$. Then there exists an efficiently sampleable B -bounded distribution χ such that if there is an efficient algorithm that solves the (average-case) $\text{dLWE}_{\ell, q, \chi}$ problem. Then:

- There is a quantum algorithm that solves SVP with approximation factor $\tilde{O}(\ell\sqrt{\ell} \cdot q/B)$ and gapSVP with approximation factor $\tilde{O}(\ell\sqrt{\ell} \cdot q/B)$ on any ℓ -dimensional lattice, and runs in time $\text{poly}(\ell)$.
- There is a classical algorithm that solves the ζ -to- γ decisional shortest vector problem $\text{gapSVP}_{\zeta, \gamma}$, where $\gamma = \tilde{O}(\ell\sqrt{\ell} \cdot q/B)$, and $\zeta = \tilde{O}(q\sqrt{\ell})$, on any ℓ -dimensional lattice, and runs in time $\text{poly}(\ell)$.

We remark that this connection is time-preserving, in the sense that given an LWE algorithm that runs in time t , these reductions produce algorithms to solve lattice problems that run in time $\text{poly}(t)$.

We refer the reader to [Reg05, Pei09] for the formal definition of these lattice problems, as they have no direct connection to this work. We only note here that the best known algorithms for these problems run in time nearly exponential in the dimension ℓ [AKS01, MV10]. More generally, the best algorithms that approximate these problems to within a factor of 2^k run in time $2^{\tilde{O}(\ell/k)}$. Specifically, given the current state of the art on lattice algorithms, the $\text{LWE}_{\ell, q, \chi}$ assumption is quite plausible for a $\text{poly}(\ell)$ -bounded distribution χ and q as large as 2^{ℓ^ϵ} (for any constant $0 < \epsilon < 1$).

Given this state of affairs, we will abuse notation slightly and conflate the LWE dimension ℓ with the security parameter κ .

B Construction of Two-Outcome Attribute-Based Encryption

Let us construct a two-outcome attribute-based encryption scheme, denoted ABE_2 , from an ABE scheme, ABE .

The idea is to use two ABE instantiations, one encrypting M_0 and the other M_1 . To make sure that exactly one of these messages gets revealed when a predicate is evaluated, we provide secret keys for the predicate and the negation of the predicate for the two instantiations.

Setup $\text{ABE}_2.\text{Setup}(1^\kappa)$:

1. Run $(\text{fmsk}_0, \text{fmpk}_0) \leftarrow \text{ABE}.\text{Setup}(1^\kappa)$ and $(\text{fmsk}_1, \text{fmpk}_1) \leftarrow \text{ABE}.\text{Setup}(1^\kappa)$.
2. Let $\text{fmsk} := (\text{fmsk}_0, \text{fmsk}_1)$ and $\text{fmpk} := (\text{fmpk}_0, \text{fmpk}_1)$. Output fmsk and fmpk .

Key generation $\text{ABE}_2.\text{KeyGen}(\text{fmsk}, P)$: Let $\text{fsk}_0 \leftarrow \text{ABE}.\text{KeyGen}(\text{fmsk}_0, \bar{P})$ and $\text{fsk}_1 \leftarrow \text{ABE}.\text{KeyGen}(\text{fmsk}_1, P)$, where \bar{P} is the negation of P , namely $\bar{P}(x) = 1 - P(x)$. Output $\text{fsk}_P = (\text{fsk}_0, \text{fsk}_1)$.

Encryption $\text{ABE}_2.\text{Enc}(\text{fmpk}, x, M_0, M_1)$: Let $C_0 \leftarrow \text{ABE}.\text{Enc}(\text{fmpk}_0, x, M_0)$ and $C_1 \leftarrow \text{ABE}.\text{Enc}(\text{fmpk}_1, x, M_1)$. Output $C = (C_0, C_1)$.

Decryption $\text{ABE}_2.\text{Dec}(\text{fsk}_P, C)$:

1. Parse $\text{fsk}_P = (\text{fsk}_0, \text{fsk}_1)$ and $C = (C_0, C_1)$.
2. Run $M_0 \leftarrow \text{ABE}.\text{Dec}(\text{fsk}_0, C_0)$ and if $M_0 \neq \perp$, output M_0 .
3. Run $M_1 \leftarrow \text{ABE}.\text{Dec}(\text{fsk}_1, C_1)$ and if $M_1 \neq \perp$, output M_1 .

We next prove that this construction yields a secure two-outcome ABE scheme. Note that our construction requires an ABE scheme where the predicate class \mathcal{P}_n is closed under negation: for every predicate $P \in \mathcal{P}_n$, the predicate \bar{P} is also included in \mathcal{P}_n .

Proof of Claim 2.5. Correctness of ABE_2 is straightforward: If $P(x) = 0$, C_0 will decrypt to M_0 by the correctness of ABE, and mutatis mutandis for $P(x) = 1$.

We prove security by contradiction. Assume there exists p.p.t. $A = (A_1, A_2, A_3)$ that breaks the security of our ABE_2 construction: Def. 2.11; namely, there exists a polynomial p such that, for infinitely many κ ,

$$\Pr[\text{Exp}_{\text{ABE}_2, A}(1^\kappa) = 1] \geq 1/2 + 1/p(\kappa). \quad (6)$$

We construct a p.p.t. adversary $R = (R_1, R_2, R_3)$ that breaks the security of ABE, Def. 2.9.

The adversary R_1 receives as input fmpk^* and outputs a predicate P^* as follows. The adversary A_1 expects two public keys. R_1 uses fmpk^* as one of these public keys and generates the other public key freshly $(\text{fmsk}, \text{fmpk}) \leftarrow \text{ABE.KeyGen}(1^\kappa)$. The order in which R_1 provides these keys to A_1 depends on the value of $P(x)$ not known at this step. If $P(x)$ will be 0, R will have to give A the ability to decrypt a ciphertext encrypted with the first key. If that key is fmpk^* , R cannot accomplish this task because it does not have the corresponding secret key. Therefore, R will try to guess $P(x)$ by flipping a random coin. Concretely, R_1 runs:

1. Guess $P(x)$ at random, namely draw a random bit denoted guess. If guess is 0:

- (a) Provide $(\text{fmpk}, \text{fmpk}^*)$ to A_1 .
- (b) Receive P from A_1 and output $P^* := P$.

2. Else [guess is 1]:

- (a) Provide $(\text{fmpk}^*, \text{fmpk})$ to A_1 .
- (b) Receive P from A_1 and output $P^* := \bar{P}$.

Adversary R_2 receives as input fsk_{P^*} and generates M_0^*, M_1^* , and x^* as follows.

1. Generate $\text{fsk}_{\bar{P}^*} \leftarrow \text{ABE.KeyGen}(\text{fmsk}, \bar{P}^*)$.
2. If guess is 0, provide $(\text{fsk}_{\bar{P}^*}, \text{fsk}_{P^*})$ to A_2 , else (guess was 1) provide $(\text{fsk}_{P^*}, \text{fsk}_{\bar{P}^*})$ to A_2 .
3. Receive (M, M_0, M_1, x) from A_2 . Output $M_0^* := M_0$, $M_1^* := M_1$ and $x^* := x$.

Adversary R_3 receives as input c^* and outputs a guess bit as follows:

1. Check that $P(x)$ equals guess. If this is not the case, namely, R_1 had guessed incorrectly the value of $P(x)$, output a random bit and exit. Otherwise, continue.
2. Feed the following input to A_3 : if guess = 0, feed inputs $(\text{ABE.Enc}(\text{fmpk}, (x, M)), c^*)$, else (guess = 1), feed inputs $(c^*, \text{ABE.Enc}(\text{fmpk}, (x, M)))$. Output whatever A_3 outputs.

R_1 guesses $P(x)$ correctly with a chance of half. When R_1 does not guess $P(x)$ correctly, R_3 outputs a correct bit with chance $1/2$ (because it outputs a random guess). When R_1 guesses $P(x)$ correctly, we can see that R simulates the ABE_2 game with A correctly. Therefore, in this case, whenever A guesses correctly, R also guesses correctly. Using Eq. (6), we have

$$\Pr[\text{Exp}_{\text{ABE},R}(1^\kappa) = 1] \geq 1/2 \cdot 1/2 + 1/2(1/2 + 1/2p(\kappa)) = 1/2 + 1/2p(\kappa), \quad (7)$$

which provides the desired contradiction. \square

C Homomorphic Encryption for Turing Machines: Definitions and Proofs

Let us first define the syntax of a Turing machine homomorphic encryption scheme.

Definition C.1. A Turing machine homomorphic encryption scheme TMFHE for a class of Turing machines \mathcal{M} is a quadruple of p.p.t. algorithms $(\text{TMFHE.KeyGen}, \text{TMFHE.Enc}, \text{TMFHE.Dec}, \text{TMFHE.Eval})$ as follows:

- $\text{TMFHE.KeyGen}(1^\kappa, 1^n, 1^{t_{\max}})$ takes as input a security parameter κ , an input size n , and a time bound t_{\max} , and outputs a public key PK, an evaluation key EVK, and a secret key SK.
- $\text{TMFHE.Enc}(\text{PK}, M, x)$ takes as input the public key PK, a Turing machine M with one bit of output, and an input $x \in \{0, 1\}^n$, for some n , and outputs a ciphertext c .
- $\text{TMFHE.Dec}(\text{SK}, c)$ takes as input the secret key SK and a ciphertext c , and outputs a message x .
- $\text{TMFHE.Eval}(\text{EVK}, c)$ takes as input the evaluation key EVK, and a ciphertext c , and outputs a ciphertext c' .

Correctness: For every polynomial $n(\cdot)$, for every polynomial $t_{\max}(\cdot)$, for every sufficiently large security parameter κ , for $n = n(\kappa)$, for every Turing machine $M \in \mathcal{M}$ with upper bound on running time for inputs of size n of $t_{\max}(n)$, and for every input $x \in \{0, 1\}^n$,

$$\begin{aligned} &\Pr[(\text{PK}, \text{EVK}, \text{SK}) \leftarrow \text{TMFHE.KeyGen}(1^\kappa, 1^n, 1^{t_{\max}(n)}); \\ &\quad c \leftarrow \text{TMFHE.Enc}(\text{PK}, M, x); \\ &\quad c^* \leftarrow \text{TMFHE.Eval}(\text{EVK}, M, c) : \\ &\quad \text{TMFHE.Dec}(\text{SK}, c^*) \neq M(x)] = \text{negl}(\kappa). \end{aligned}$$

Note that the correctness property constraints t_{\max} to be a polynomial. However, t_{\max} can still be a very large polynomial and we would like the server to not have to run in that time for all inputs. (In fact, this constraint can be eliminated if we use a FHE scheme and an ABE scheme that have no correctness error).

Definition C.2 (Runtime-CPA Security). Let TMFHE be an input-specific homomorphic encryption scheme for the class of Turing machines \mathcal{M} . For every p.p.t. adversary $A = (A_1, A_2)$ and p.p.t. simulator S , consider the following two experiments:

$$\text{Exp}_{\text{TMFHE},A}^{\text{real}}(1^\kappa) :$$

- 1: $(1^{t_{\max}}, 1^n, \text{state}_A) \leftarrow A_1(1^\kappa)$.
- 2: $(\text{PK}, \text{EVK}, \text{SK}) \leftarrow \text{TMFHE.KeyGen}(1^\kappa, 1^n, 1^{t_{\max}})$
- 3: $(M, x, \text{state}'_A) \leftarrow A_2(\text{state}_A, \text{PK}, \text{EVK})$
- 4: $c \leftarrow \text{TMFHE.Enc}(\text{PK}, M, x)$
- 5: Output (state'_A, c)

$$\text{Exp}_{\text{TMFHE},A,S}^{\text{ideal}}(1^\kappa) :$$

- 4: $\tilde{c} \leftarrow S(M, 1^n, 1^t, \text{EVK}, \text{PK})$ with $t = \text{time}(M, x)$
 - 5: Output $(\text{state}'_A, \tilde{c})$
-

The scheme is said to be runtime-CPA-secure if there exists a p.p.t. simulator S such that for all pairs of p.p.t. adversaries $A = (A_1, A_2)$ for which A_2 outputs $M \in \mathcal{M}$ and $x \in \{0, 1\}^n$, we have

$$\left\{ \text{Exp}_{\text{TMFHE},A}^{\text{real}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}} \stackrel{c}{\approx} \left\{ \text{Exp}_{\text{TMFHE},A,S}^{\text{ideal}}(1^\kappa) \right\}_{\kappa \in \mathbb{N}}.$$

This definition essentially captures our security goal: one can simulate any information learned from the scheme by using only the Turing machine M and the running time of M on x , but without any other information about x .

In fact, we can achieve a scheme that hides M as well in a straightforward way: since our construction passes M and x as inputs to universal circuits, M could also be hidden in the same way as x is.

C.1 Proof

Proof of Theorem 6.2. We first prove the correctness and efficiency claims of the theorem and then we prove security.

If the underlying FE scheme is correct, then TMFHE is correct; whenever 2^i for some i is an upper bound on the running time of M on x , then $C_i(M, x)$'s output is 1. Based on the correctness of the FHE scheme FHE, the evaluation of D_i on \hat{M}, \hat{x} will be correct, so FHE.Dec will return $M(x)$.

Lemma C.1. *The online work of the client in the TMFHE scheme is $(\log t_{\max} + n) \cdot \text{poly}(\kappa, d(n))$.*

Proof. The work of the client in the online phase consists of running $\text{TMFHE.Enc}(\text{PK}, x)$ and $\text{TMFHE.Dec}(\text{SK}, c)$. The work of the client for $\text{TMFHE.Enc}(\text{PK}, x)$ is $n \text{poly}(d(\kappa))$ to compute the FHE ciphertexts and $(1 + \lceil \log t_{\max} \rceil) \cdot \text{poly}(d(n), \kappa)$ to compute the FE ciphertexts. Since n depends polynomially in κ , we obtain that total cost is at most $(\log t_{\max} + n) \text{poly}(\kappa, d(n))$ (where we incorporated the constant values in the poly notation).

The runtime of $\text{TMFHE.Dec}(\text{SK}, c)$ is $\text{poly}(d(\kappa))$ because FHE.Enc runs polynomial in κ and $d(\kappa)$. Therefore, the total online work of the client is $(\log t_{\max} + n) \text{poly}(d(\kappa), d(n), \kappa)$. \square

Lemma C.2. *The work of the evaluator in the TMFHE scheme is $\text{poly}(n, d(n), \text{time}(M, x))$.*

Proof. The work of the evaluator consists of running $\text{TMFHE.Eval}(\text{EVK}, M, c)$. This depends on the number of times the loop in TMFHE.Eval is repeated and the cost within each loop. Let us evaluate the cost at the i -th repetition of the loop. Let $t_i = 2^i$.

By the properties of the transformation \mathcal{T}_n , the size of C_i is at most $t_i \text{polylog } t_i$. The cost of evaluating $\text{FE.Dec}(\text{fsk}_i, c_i)$ is therefore $\text{poly}(n, d(n), t_i \text{polylog } t_i)$.

If t is the runtime of M on x , the index i at which the loop will halt (because the evaluator obtained a value the bit b being one) is at most $1 + \lceil \log t \rceil$. Therefore, the loop will repeat at most $1 + \lceil \log t \rceil$ times.

$$\begin{aligned} \text{Runtime of TMFHE.Eval(EVK, } c) &= \sum_{i=1}^{1+\lceil \log t \rceil} \text{poly}(n, d(n), t_i \text{ polylog } t_i) \\ &\leq (1 + \lceil \log t \rceil) \text{poly}(n, d(n), t \text{ polylog } t) \\ &\leq \text{poly}(n, d(n), t \text{ polylog } t) = \text{poly}(n, d(n), t), \end{aligned}$$

where the last equality comes from adjusting the implicit polynomial in poly . Note that even though EVK consists of $\log t_{\max}$ such fsk_i keys, TMFHE.Eval does not have to read all of EVK. \square

Finally, we prove security of the scheme.

Lemma C.3. *The TMFHE protocol is runtime-CPA-secure.*

Proof. To prove that our TMFHE construction is secure, we provide a simulator S , as in Def. C.2. The simulator S invokes the simulator of the functional encryption scheme, as in Def. 2.13, which we denote Sim_{FE} . The simulator S receives inputs $M, 1^n, 1^t, \text{EVK}$, and PK , and proceeds as follows:

1. Compute $\hat{0}^n \leftarrow (\text{FHE.Enc}(\text{hpk}, 0), \dots, \text{FHE.Enc}(\text{hpk}, 0))$ (n times).
2. For each $i \in [\tau]$, compute the circuits $D_i = T_n(2^i)$ and then C_i as before; we remind the reader that C_i , on input a TM M and a value x , outputs 1 if M finished in 2^i steps when running on input x or 0 otherwise.
3. For each i such that $2^i < t$:
 - (a) Call the simulator Sim_{FE} to simulate a computation result of 0 because M could not have finished its computation at step i . Specifically, compute $\tilde{c}_i \leftarrow \text{Sim}_{\text{FE}}(\text{fmpk}_i, \text{fsk}_i, C_i, 0, 1^{n+|M|})$.
4. For each i such that $2^i \geq t$:
 - (a) Call the simulator Sim_{FE} to simulate an answer of 1 because M finished computation on the input (unknown to S). Thus, compute $\tilde{c}_i \leftarrow \text{Sim}_{\text{FE}}(\text{fmpk}_i, \text{fsk}_i, C_i, 1, 1^{n+|M|})$.
5. Output $\tilde{c} = (\hat{0}, \tilde{c}_1, \dots, \tilde{c}_\tau)$.

To prove that S satisfies Def. C.2, we use three hybrids:

Hybrid 0: The ideal experiment with simulator S .

Hybrid 1: The same as Hybrid 0 but $\hat{0}^n$ gets replaced with $\hat{x} = (\text{FHE.Enc}(\text{hpk}, x_1), \dots, \text{FHE.Enc}(\text{hpk}, x_n))$.

Hybrid 2: The real experiment.

It is easy to see that the outcome of Hybrid 0 and the outcome of Hybrid 1 are computationally indistinguishable because FHE is semantically secure: the encryptions of 0^n in Hybrid 0 and the encryption of x in Hybrid 1 are both generated with fresh randomness, and the secret key hsk (or any function of hsk other than a fresh encryption) is never released to any adversary.

Now let us look at Hybrid 1 and Hybrid 2. These are computationally indistinguishable based on a standard hybrid argument invoking the security of Sim_{FE} as follows.

Multi-Input Functional Encryption

Shafi Goldwasser* Vipul Goyal† Abhishek Jain‡ Amit Sahai§

Abstract

We introduce the problem of Multi-Input Functional Encryption, where a secret key SK_f can correspond to an n -ary function f that takes multiple ciphertexts as input. Multi-input functional encryption is a general tool for computing on encrypted data which allows for *mining aggregate information from several different data sources* (rather than just a single source as in single input functional encryption). We show wide applications of this primitive to running SQL queries over encrypted database, non-interactive differentially private data release, delegation of computation, etc.

We formulate both indistinguishability-based and simulation-based definitions of security for this notion, and show close connections with indistinguishability and virtual black-box definitions of obfuscation. Assuming indistinguishability obfuscation for circuits, we present constructions achieving indistinguishability security for a large class of settings. We show how to modify this construction to achieve simulation-based security as well, in those settings where simulation security is possible. Assuming differing-inputs obfuscation [Barak et al., FOCS'01], we also provide a construction with similar security guarantees as above, but where the keys and ciphertexts are *compact*.

*MIT and Weizmann. shafi@csail.mit.edu

†Microsoft Research, India. vipul@microsoft.com

‡Boston University and MIT. abhishek@csail.mit.edu

§UCLA. sahai@cs.ucla.edu

The simulator Sim_{FE} is called τ times. Let $\tilde{c}_i^{(1)}$ be the ciphertext output by the simulator for the i -th invocation in Hybrid 1, and let c_i be the ciphertext output in Hybrid 2 on the i -th invocation. It is enough to prove that the outcome of these two experiments consisting of state'_A and only one of the ciphertexts (e.g., $\tilde{c}_i^{(1)}$ or c_i) are computationally indistinguishable. The reason is that one can employ a standard hybrid argument consisting of $\tau + 1$ sub-hybrids, the 0-th sub-hybrid being Hybrid 1 and the τ -th sub-hybrid being Hybrid 2 and any intermediary sub-hybrid i has the first i ciphertexts as in Hybrid 2 and the rest as in Hybrid 1. Such an argument is possible because τ is polynomial in the security parameter and each ciphertext is encrypted with independently generated public keys.

Therefore, all we need to argue is that the outcome of Hybrid 1 and Hybrid 2 consisting of state'_A and only $\tilde{c}_i^{(1)}$ (c_i respectively) are computationally indistinguishable. This follows directly because Sim_{FE} satisfies the FULL-SIM-secure functional encryption definition, Def. 2.13. \square

The three lemmas above complete the proof of the theorem. \square

1 Introduction

Traditionally, encryption has been used to secure a communication channel between a unique sender-receiver pair. In recent years, however, our networked world has opened up a large number of new usage scenarios for encryption. For example, a single piece of encrypted data, perhaps stored in an untrusted cloud, may need to be used in different ways by different users. To address this issue, the notion of functional encryption (FE) was developed in a sequence of works [SW05, GPSW06, BW07, KSW08, LOS⁺10, BSW11, O’N10]. In functional encryption, the owner of the master secret key MSK can create a secret key SK_f for any function f from a family \mathcal{F} . Given any ciphertext CT with underlying plaintext x , using SK_f a user can efficiently compute $f(x)$. The security of FE requires that the adversary “does not learn anything” about x , other than the computation result $f(x)$.

How to define “does not learn anything about” x is a fascinating question which has been addressed by a number of papers, with general formal definitions first appearing in [BSW11, O’N10]. The definitions range from requiring a strict *simulation* of the view of the adversary, which enlarges the range of applications, but has been shown to either necessitate a secret key whose size grows with the number of ciphertexts that will ever be released [BSW11, BO13] (or a ciphertext whose size grows with the number of functions for which secret keys will ever be released [AGVW13, CIJ⁺13]), to an *indistinguishability* of ciphertexts requirement which supports the release of an unbounded number of function keys and ciphertexts.

Functional encryption seems to offer the perfect non-interactive solution to many problems which arise in the context of delegating services to outside servers. A typical example is the delegation of spam filtering to an outside server as follows: Alice publishes her public key online and gives the spam filter a key for the filtering function; users sending email to Alice will encrypt the email with her public key. The spam filter can now determine by itself, for each email, whether to pass it along to Alice’s mailbox or to deem it as spam, but without ever learning anything else about Alice’s email. This example inherently requires computing a function f on a *single* ciphertext.

Multi-Input Functional Encryption. It is less clear, however, how to define or achieve functional encryption in the context of computing a function defined over *multiple* plaintexts given their corresponding ciphertexts, or further, given their ciphertexts each encrypted under a different key. Yet, these settings, which we formalize as *Multi-Input Functional Encryption* (MI-FE), encompass a vast landscape of applications, going way beyond delegating computation to an untrusted server or cloud. Multi-input functional is a very general tool for computing on encrypting data, which allows for *mining aggregate information from several different data sources* (rather than just a single source as in single input functional encryption).

Let us begin by clarifying the setting of Multi-Input Functional Encryption: Let f be an n -ary function where $n > 1$ can be a polynomial in the security parameter. In MI-FE, the owner of a master secret key MSK can derive special keys SK_f whose knowledge enables the computation of $f(x_1, \dots, x_n)$ from n ciphertexts $\text{CT}_1, \dots, \text{CT}_n$ of underlying messages x_1, \dots, x_n with respect to the same master secret key MSK . We allow the different ciphertexts c_i to be each encrypted under a *different* encryption key EK_i to capture the setting in which each ciphertext was generated by an entirely different party.

Let us illustrate a few settings that illustrate the applicability of MI-FE.

Example 1: Running SQL Queries on Encrypted Database. Suppose we have an encrypted database. A natural goal in this scenario would be to allow a party Alice to perform a certain class of general SQL queries over this database (e.g., Alice may only be authorized to access records created on a certain date). If we use ordinary functional encryption, Alice would need to obtain a separate secret key for every possible valid SQL query, a potentially exponentially large set. Multi-input functional encryption allows us to address this problem in

a flexible way. We highlight two aspects of how MI-FE can apply to this example:

- Let f be the function where $f(Q, x)$ first checks if Q is a valid SQL query from the allowed class, and if so $f(Q, x)$ is the output of the query Q on the database x . Now, if we give the secret key SK_f and the encryption key EK_1 to Alice, then Alice can choose a valid query Q and encrypt it under her encryption key EK_1 to obtain ciphertext CT_1 . Then she could use her secret key SK_f on ciphertexts CT_1 and CT_2 , where CT_2 is the encrypted database, to obtain the results of the SQL query.
- Furthermore, if the database is dynamic (rather than static) with individual entries being added, modified, or, deleted, the most natural way to build such a database would be to have different ciphertexts for each entry in the database. In this case, for a database of size n , we could let f be an $(n + 1)$ -ary function where $f(Q, x_1, \dots, x_n)$ is the result of a (valid) SQL query Q on the database (x_1, \dots, x_n) .

Example 2: Computing over Encrypted Data Stream. Suppose ciphertexts correspond to a stream of encrypted phone calls (or video frames produced by surveillance cameras), produced separately by several different devices. Law enforcement agencies may require the ability to run algorithms which check the calls or videos for suspicious activities (these algorithms need to analyze sequences of calls or frames rather than individual calls/frames) in which case (and only in this case) court orders can be obtained to decrypt the phone calls (or videos) in their entirety. Here, the need is to compute a function $f(p_1, \dots, p_n)$ where p_i is the i 'th phone call, encrypted to form the ciphertext c_i .

More generally, suppose ciphertexts c_1, \dots, c_n correspond to a *list* of encrypted inputs to some algorithm, e.g. a list of edges x_1, \dots, x_n in a graph for a routing algorithm f . Then, we need to run the algorithm $f(x_1, \dots, x_n)$ across multiple ciphertexts. It is likely that this type of algorithm would be the rule rather than the exception in the context of algorithms run over large inputs.

Example 3: Non-Interactive Differentially Private Data Release. Suppose there are several hospitals each of which holds a collection of individual blood samples. They would like to participate in clinical trials performed by various researchers. The hospitals cannot simply release the blood samples records because of various patient privacy laws. However, the hospitals are willing to allow a clinical study researcher to compute an aggregate function f over multiple samples x_i to learn $y = f(x_1, \dots, x_n)$ as long as f achieves a sufficient level of privacy.

While such a scenario is addressed by differential privacy [DMNS06], existing solutions require each hospital to interact with the researcher in every trial (potentially via a multi-party computation protocol when several hospitals are involved). Indeed, it is known that non-cryptographic methods for allowing the hospitals to *non-interactively* prepare their records in a way that would later allow for meaningful and diverse research studies *must* incur high accuracy loss [DNR⁺09].

Multi-input functional encryption can address this problem by having the hospitals encrypt the samples x_i to obtain ciphertexts CT_i , and publish all the ciphertexts. This step can be performed by the hospitals non-interactively *before* any research trial f is decided (in contrast to the standard differential-privacy setting where f is decided upon first and then the “differentially private” information collection algorithm takes place). Later, a researcher who wishes to compute an algorithm f' (that is guaranteed to provide sufficient privacy) would be given a secret key $SK_{f'}$ (potentially by a trusted agency such as the government) that she can use to obtain the output of her algorithm on the blood samples. In this manner, we can obtain high accuracy while still guaranteeing good (computational) privacy.

We remark that this example requires MI-FE to support *randomized* functionalities. Our positive results, discussed later, handle this case.

Example 4: Multi-client Delegation of Computation. In a multi-client delegation scheme

[CKKC13], multiple weak clients C_1, \dots, C_n wish to jointly delegate the computation of an n -ary function f on their inputs x_1, \dots, x_n to a computationally powerful server. The efficiency requirement of a delegation scheme is that the computation of the clients should be independent of the size of f . From a security viewpoint, we require that a dishonest server should not be able to convince the (honest) clients on an incorrect output.

Multi-input functional encryption provides a natural solution to this problem similar to how single-input functional encryption provides a solution for *single-client* delegation of computation [PRV12, GVW13, GGH⁺13b, GKP⁺13b, GGH⁺13a]. Details of this are provided in Section 1.1.3.

Our Goal. As these examples illustrate, extending the scope of functional encryption to address functions defined over multiple ciphertexts can be highly beneficial. In short, it could provide a non-interactive method to compute n -ary functions on encrypted inputs (possibly by different parties), analogously to interactive multi-party secure computations defined over multiple inputs held by n different parties.

Extending functional encryption to address the multi-input setting is the focus of this work.

1.1 This paper

This paper is dedicated to the study of multi-input functional encryption, starting with formalizations of security. We provide both feasibility results and negative results with respect to different definitions of security. Following the single-input setting, we consider two notions of security, namely, indistinguishability-based security (or IND security for short) and simulation-based security (or SIM security for short).

1.1.1 Indistinguishability-based Security

We start by considering the notion of indistinguishability-based security for functional encryption for n -ary functions: Informally speaking, in IND security for MI-FE, we consider a game between a judge and an adversary. First, the judge generates the master secret key MSK , n encryption keys $\{\text{EK}_1, \dots, \text{EK}_n\}$ and gives to the adversary a subset of the encryption keys (chosen by the adversary). Then the adversary can request any number of secret keys SK_f for functions f of her choice. Next, the adversary declares two “challenge vectors” \vec{X}^0 and \vec{X}^1 , where every $X_i^b \in \vec{X}^b$ is a set of plaintexts $\{x_{i,1}^b, \dots, x_{i,n}^b\}$. The judge chooses a bit b at random, and for each $j \in [n]$, the judge encrypts every element $x_{i,j}^b$ of X_i^b (for every i) using encryption key EK_j to obtain a tuple of “challenge ciphertexts” $\vec{\text{CT}}$, which is given to the adversary. After this, the adversary can again request any number of secret keys SK_f for functions f of her choice. Finally, the adversary has to guess the bit b that the judge chose.

If the adversary has requested a secret key for any function f such that there exist *splitting* input vectors \vec{y}^0 and \vec{y}^1 that satisfy the following two properties:

1. For every $j \in [n]$, either $\exists i$ such that $y_j^b \in X_i^b$ or the adversary has EK_j , and
2. $f(\vec{y}^0) \neq f(\vec{y}^1)$,

then the adversary loses the game – because the legitimate functionalities that he has access to already allow him to distinguish between the scenario where $b = 0$ and $b = 1$. If the adversary never queries a secret key for such a function but nevertheless guesses b correctly, we say that she wins. The IND security definition requires that the adversary’s probability of winning be at most negligibly greater than $\frac{1}{2}$.

This definition generalizes the indistinguishability-based definition of (single-input) functional encryption, which was historically the first security notion considered for functional encryption [SW05]. Informally speaking, this definition captures an information-theoretic flavor of

security, where the adversary should not learn anything beyond what is information-theoretically revealed by the function outputs it can obtain.

With regards to IND-secure MI-FE, we obtain the following results:

IND-secure MI-FE from Indistinguishability Obfuscation. Assume the existence of an indistinguishability obfuscator [BGI⁺01] for general circuits (the first candidate construction for the same was recently put forward by [GGH⁺13a]) and one-way functions, we provide a construction for IND-secure MI-FE for general circuits for any polynomial-size challenge vectors, with any subset of encryption keys given to the adversary. Furthermore, our construction has security when the adversary can obtain any *unbounded* polynomial number of secret keys SK_f . We prove the security in the selective model, where the adversary must begin by declaring the challenge vectors. By using complexity leveraging (and thereby assuming sub-exponentially secure indistinguishability obfuscation and sub-exponentially secure one-way functions, we can achieve full security in a standard manner.

Compact IND-secure MI-FE from Differing-Inputs Obfuscation. Our first construction only supports challenge vectors with an a priori fixed (polynomial) size q . In particular, the size of the encryption keys and ciphertexts in the scheme grows with q . Towards this end, assuming the existence of the stronger notion of differing-inputs obfuscation [BGI⁺01] and one-way functions, we provide a second construction for IND secure MI-FE with “compact” keys and ciphertexts, i.e., the size of the keys and ciphertexts in the scheme is *independent* of q . Further, we directly prove *full* security of our scheme against adversaries that know any subset of encryption keys and an unbounded polynomial number of secret keys SK_f .

IND-secure MI-FE implies Indistinguishability Obfuscation. Finally, we show that the existence of IND-secure MI-FE for general circuits implies the existence of an indistinguishability obfuscator for general circuits, even when:

1. The MI-FE scheme is only secure against adversaries that can obtain a *single* secret key.
2. The adversary does not know any encryption keys, i.e., the MI-FE scheme is a *secret-key* scheme.

This stands in stark contrast to the single-input setting, where [SS10] showed how to obtain single-key secure (single input) functional encryption for all circuits, under only the assumption that public-key encryption exists. Indeed, further research in single-key security for functional encryption has largely focused on efficiency issues [GKP⁺13b, GKP⁺13a] such as succinctness of ciphertexts, that enable new applications. In the setting of multi-input security, in contrast, even single key security must rely on the existence of indistinguishability obfuscation.

1.1.2 Simulation-based security

In simulation-based security, informally speaking, we require that every adversary can be simulated using only oracle access to the functions f for which the adversary obtains secret keys, even when it can obtain a set of “challenge” ciphertexts corresponding to unknown plaintexts – about which the simulator can only learn information by querying the function f at these unknown plaintexts. We highlight two natural settings for the study of SIM-secure MI-FE: (1) the setting where an adversary has access to an encryption key (analogous to the public-key setting), and (2) the setting where the adversary does not have access to any encryption keys (analogous to the secret key setting). The security guarantees which are achievable in these settings will be vastly different as illustrated below.

Several works [BSW11, AGVW13, BO13, CIJ⁺13] have shown limitations on parameters with respect to which SIM security can be achieved for single-input functional encryption. For

multi-input functional encryption, due to the connection to obfuscation discussed above, the situation for **SIM** security is more problematic. We provide the following results for **SIM**-secure MI-FE:

SIM-secure MI-FE implies Virtual Black-Box Obfuscation. We first show that **SIM**-secure MI-FE implies virtual black-box (VBB) obfuscation in various settings. Specifically, we show:

1. If there exists a *secret-key* MI-FE scheme for general circuits that achieves **SIM** security against adversaries that request: (a) a *single* key for a general function f and (b) a set of challenge ciphertexts that can (informally speaking) form a *super-polynomial* number of potential inputs to f , then VBB obfuscation must be possible for general circuits.
2. If there exists an MI-FE scheme for 2-ary functions that achieves **SIM** security against adversaries that request: (a) a *single* key for a 2-ary function, and (b) *one* of the two encryption keys and *one* challenge ciphertext, then VBB obfuscation must be possible for general circuits.

Since VBB obfuscation is known to be impossible for general circuits [BGI⁺01], this yields us impossibility results for **SIM**-secure MI-FE beyond those known in the single-input setting. See Section 6 for details.

SIM-secure Secret-Key MI-FE against Unbounded Collusions. In light of these negative results, the only hope for obtaining a positive result lies in a situation where: (a) *no* encryption keys are given to the adversary, and (b) the challenge ciphertexts given to the adversary can only form a *polynomial* number of potential inputs to valid functions.

Towards this end, assuming one-way functions and indistinguishability obfuscation, for any fixed polynomial bound q on the size of challenge plaintexts, we give a construction for **SIM**-secure secret-key MI-FE for general circuits against adversaries that can obtain an unbounded polynomial number of secret keys SK_f *after* obtaining the challenge ciphertexts. The size of the encryption keys and ciphertexts in this scheme grows with q .

We also provide another construction based on one-way functions and differing-inputs obfuscation that achieves the same security guarantees as above. The encryption keys and ciphertexts in this scheme are “compact”, i.e., their sizes are independent of q .

1.1.3 Extensions and Applications

MI-FE for Randomized Functions. Very recently, Goyal et al. [GJKS13] first studied the question of constructing single-input functional encryption schemes for *randomized* functionalities. By building on their techniques, we show how to extend our positive results to handle general n -ary randomized functionalities. In particular, this allows us to obtain a non-interactive computationally differentially private mechanism, as discussed earlier.

MI-FE for Turing Machines. The problem of single-input functional encryption for turing machines was first studied by Goldwasser et al. [GKP⁺13a]. Very recently, Boyle et al. [BCP13] and Ananth et al. [ABG⁺13] provide constructions of single-input functional encryption for turing machines against an unbounded polynomial number of key queries. We observe that their techniques can be leveraged to extend our results to MI-FE for turing machines, thereby achieving *input-specific running times*. The resulting construction would inherit from these works the underlying assumptions of differing-inputs obfuscation, succinct non-interactive argument of knowledge (SNARK) [BCCT12], fully-homomorphic encryption [Gen09] and collision-resistant hash functions. We omit the details from this manuscript and refer the reader to [ABG⁺13, BCP13].

Hierarchical MI-FE. The notion of hierarchical identity-based and attribute-based encryption is well studied in the literature (see e.g., [GS02, BW06, LOS⁺10]). In the context of (single-input) functional encryption, this problem is stated as follows: We require that the owner of a secret key SK_f can derive new keys corresponding to any function g that can be defined as a composition of f' on f (i.e., $f' \circ f$) for some function f' .

Recently, [ABG⁺13] observe that the construction [GGH⁺13a] is already flexible enough to yield a hierarchical (single-input) functional encryption scheme. We note that the same ideas carry over to our constructions of MI-FE. We refer the reader to [ABG⁺13] for details.

Multi-Client Delegation of Computation. Here we briefly discuss how an MI-FE scheme provides a solution for multi-client delegation of computation. We follow the approach of Parno et al. [PRV12], adapted to the multi-client setting. Given an MI-FE scheme, the clients first participate in a pre-processing phase where they jointly compute two pairs of master secret and encryption keys $(\text{MSK}_1, \text{EK}_1)$, $(\text{MSK}_2, \text{EK}_2)$ and random values (r_1, r_2) . Let f be the function that the clients wish to delegate. The clients use MSK_1 to compute a secret key SK_g for a function g that takes as input n tuples $(x_1, r), \dots, (x_n, r)$ and outputs r if $f(x_1, \dots, x_n) = 1$. Similarly, the clients use MSK_2 to compute a secret key $\text{SK}_{\bar{g}}$ for the function \bar{g} that is the same as g except that it outputs r if $f(x_1, \dots, x_n) = 0$. While these are computationally expensive operations, note that this phase is executed only *once*. The keys SK_g and $\text{SK}_{\bar{g}}$ are sent over to the worker.

Later, in an “online” phase, when the clients wish to compute f on a set of inputs x_1, \dots, x_n , each client C_i sends over encryption of (x_1, r_1) under key MPK_1 and (x_1, r_2) under MPK_2 to the worker. Now, from the properties of the MI-FE scheme, it follows that if $f(x_1, \dots, x_n) = 1$, then the server would obtain r_1 using SK_g and \perp using $\text{SK}_{\bar{g}}$ and no information about r_2 (and vice-versa, if $f(x_1, \dots, x_n) = 0$). Thus, r_1 provides a proof of the fact that the function output is 1.¹

The main advantage of this approach is that the online phase is *non-interactive*: each client can execute the online phase independently of the other clients, without any interaction.

1.1.4 Our Techniques

We have several results in this work, but to provide a flavor of the kind of difficulties that arise in the MI-FE setting, we now discuss some of the issues that we deal with in the context of our positive result for IND-secure MI-FE. (We note that similar issues arise in our positive results for SIM-secure MI-FE.)

The starting point for our construction and analysis is the recent single-input functional encryption scheme for general circuits based on indistinguishability obfuscation due to [GGH⁺13a]. However, the central issue that we must deal with is one that does not arise in their context: Recall that in the indistinguishability security game, the adversary is allowed to get secret keys for any function f , as long as this function does not “split” the challenge vectors \vec{X}^0 and \vec{X}^1 . That is, as long as it is *not* the case that there exist vectors of plaintexts \vec{x}^0 and \vec{x}^1 where for every $i \in [n]$, either there exists j such that $x_i^b \in X_j^b$ or the adversary has EK_i , such that $f(x^0) \neq f(x^1)$. A crucial point here is what happens for an index i where the adversary does *not* have EK_i . Let us consider an example with a 3-ary function, where the adversary has EK_1 , but neither EK_2 nor EK_3 .

Suppose the challenge ciphertexts $(\text{CT}_1, \text{CT}_2, \text{CT}_3)$ are encryptions of either (y_1^0, y_2^0, y_3^0) or (y_1^1, y_2^1, y_3^1) . Now, any function f that the adversary queries is required to be such that $f(\cdot, y_2^0, y_3^0) \equiv f(\cdot, y_2^1, y_3^1)$ and $f(y_1^0, y_2^0, y_3^0) = f(y_1^1, y_2^1, y_3^1)$. However, there may exist an input plaintext (say) z such that $f(y_1^0, y_2^0, z) \neq f(y_1^1, y_2^1, z)$. This is not “supposed” to be a problem because the adversary does not have EK_3 , and therefore it cannot actually query f with z as its third argument.

¹We note that this solution easily extends to functions with multi-bit outputs. See [PRV12] for details.

However, in the obfuscation-based approach to functional encryption of [GGH⁺13a] that we build on, the secret key for f is essentially built on top of an obfuscation of f . Let CT^* denote an encryption of z w.r.t. EK_3 . Then, informally speaking, in one of our hybrid experiments, we will need to move from an obfuscation that on input $(\text{CT}_1, \text{CT}_2, \text{CT}^*)$ would yield the output $f(y_1^0, y_2^0, z)$ to another obfuscation that on the same input would yield the output $f(y_1^1, y_2^1, z)$. Again, while an adversary may not be able explicitly perform such a decryption query, since we are building upon *indistinguishability obfuscation* – which only guarantees that obfuscations of circuits that implement *identical* functions are indistinguishable – such a hybrid change would not be indistinguishable since we know that $f(y_1^0, y_2^0, z) \neq f(y_1^1, y_2^1, z)$ are not identical. (We remark that we must address this issue even when using differing-inputs obfuscation in order to obtain a formal contradiction.)

Solving this problem is the core technical aspect of our constructions and their analysis. At a very high level, we address this problem by introducing a new “flag” value that can change the nature of the function f that we are obfuscating to “disable” all plaintexts except for the ones that are in the challenge vectors. We describe the details and our analysis in Section 4.

1.2 Related Works

Single-input Functional Encryption. The notion of (single-input) functional encryption was developed in a sequence of works [SW05, GPSW06, BW07, KSW08, LOS⁺10, BSW11, O’N10]. For general functions, [SS10] first showed how to obtain single-key SIM-secure FE based on standard public-key encryption. Gorbunov et al [GVW12] showed how to obtain SIM-secure FE for general circuits for a polynomially bounded number of (non-adaptive) key queries, based on public-key encryption and pseudorandom generators in NC^1 . Goldwasser et al. [GKP⁺13b] improved this result to obtain constructions with “compact” ciphertexts based on sub-exponential learning with errors assumption. Garg et al. [GGH⁺13a] construct an IND-secure FE scheme based on indistinguishability obfuscation and one-way functions, that supports an unbounded polynomial number of ciphertexts and key queries. Combining their result with [CIJ⁺13], one can obtain SIM-secure FE for general circuits supporting an unbounded number of (adaptive) key queries.

Goldwasser et al. [GKP⁺13a] give a construction of an FE scheme for turing machines based on extractable witness encryption [GGSW13] and SNARK [BCCT12]. Recently, the works of Boyle et al. [BCP13] and Ananth et al. [ABG⁺13] provide constructions of functional encryption for turing machines, supporting an unbounded number of key queries. Both of these results rely on the notion of differing-inputs obfuscation, introduced by Barak et al. [BGI⁺01] (and some other assumptions; see Section 1.1.3). We note that our usage of differing-inputs obfuscation is very similar to [BCP13, ABG⁺13].

Order-Preserving Encryption. The notion of order-preserving encryption was introduced by Boldyreva et al. [BCLO09]. Very roughly, in an order-preserving encryption scheme, for any two plaintexts x_1 and x_2 such that $x_1 > x_2$, the encryptions of x_1 and x_2 must also satisfy the same order relationship. Thus, given two ciphertexts CT_1 and CT_2 , one can simply compare them to (publicly) determine the order relationship between their underlying plaintexts.

Positive results for order-preserving encryption were given by [BCLO09, BCO11]. These results, however, achieve very weak security guarantees (in particular, they show that an order-preserving encryption scheme cannot achieve IND security). We note that one can cast the problem of computing order relationships between (encrypted) plaintexts as multi-input function encryption for comparison functionality. Specifically, instead of requiring that ciphertexts obey the same order relationship as their underlying plaintexts, we can now release secret keys to enable the computation of order relationship between encrypted plaintexts. This allows us to achieve IND security as well as SIM security, both of which provide much stronger guarantees

than [BCLO09, BCO11]. Indeed, achieving stronger security guarantees in this context was left as an open problem by [BCLO09, BCO11].

Property-Preserving Encryption. Recently, Pandey and Rouselakis [PR12] studied the problem of property-preserving encryption as a generalization of order-preserving encryption. As above, we note that this problem can be viewed as a multi-input functional encryption, where the function family is determined by the class of properties that one wishes to support. Again, we note that the security definitions considered in [PR12] are weaker than what we consider in this work. In particular, this is because we do not require the ciphertexts to satisfy the same property as their underlying plaintexts; instead in our setting, given a secret key SK_f for a property f , one can test f on the plaintexts via a joint decryption of the corresponding ciphertexts.

1.3 Organization

The rest of this paper is organized as follows. We start by presenting our definitions for multi-input functional encryption in Section 2. Next, in Section 3, we recall the definitions for various cryptographic primitives used in our constructions. We then present our constructions for multi-input functional encryption in Section 4 and Section 5. In Section 6, we show how to construct general obfuscation from multi-input functional encryption and also provide impossibility results for SIM-secure MI-FE. Finally, we discuss how to extend our positive results to handle randomized functionalities in Section 7.

2 Multi-Input Functional Encryption

In this work, we study functional encryption for n -ary functions, where $n > 1$ (and in general, a polynomial in the security parameter). In other words, we are interested in encryption schemes where the owner of a “master” secret key can generate special keys SK_f that allow the computation of $f(x_1, \dots, x_n)$ from n ciphertexts $\text{CT}_1, \dots, \text{CT}_n$ corresponding to messages x_1, \dots, x_n , respectively. We refer to such an encryption scheme as *multi-input* functional encryption. Analogously, we will refer to the existing notion of functional encryption (that only considers single-ary functions) as *single-input* functional encryption.

Intuitively, while single-input functional encryption can be viewed as a specific (non-interactive) way of performing two-party computation, our setting of multi-input functional encryption captures *multiparty* computation. Going forward with this analogy, we are interested in modeling the general scenario where the n input ciphertexts are computed by n *different* parties. This raises the following two important questions:

1. Do the parties (i.e., the encryptors) share the *same* encryption key or do they use *different* encryption keys EK_i to compute input ciphertexts CT_i .
2. Are the encryption keys secret or public?

As we shall see, these questions have important bearing on the security guarantees that can be achieved for multi-input functional encryption.

Towards that end, we present a general, unified syntax and security definitions for multi-input functional encryption. We consider encryption systems with n encryption keys, some of which may be public, while the rest are secret. When all of the encryption keys are public, then this represents the “public-key” setting, while when all the encryption keys are secret, then this represents the “secret-key” setting. Looking ahead, we remark that our modeling allows us to capture the intermediary cases between these two extremes that are interesting from the viewpoint of the security guarantees possible.

The rest of this section is organized as follows. We first present the syntax and correctness requirements for multi-input FE in Section 2.1). Then, in Section 2.2, we present our security definitions for multi-input FE.

2.1 Syntax

Throughout the paper, we denote the security parameter by k . Let $\mathcal{X} = \{\mathcal{X}_k\}_{k \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_k\}_{k \in \mathbb{N}}$ be ensembles where each \mathcal{X}_k and \mathcal{Y}_k is a finite set. Let $\mathcal{F} = \{\mathcal{F}_k\}_{k \in \mathbb{N}}$ be an ensemble where each \mathcal{F}_k is a finite collection of n -ary functions. Each function $f \in \mathcal{F}_k$ takes as input n strings x_1, \dots, x_n , where each $x_i \in \mathcal{X}_k$ and outputs $f(x_1, \dots, x_n) \in \mathcal{Y}_k$.

A multi-input functional encryption scheme \mathcal{FE} for \mathcal{F} consists of four algorithms (FE.Setup, FE.Enc, FE.Keygen, FE.Dec) described below.

- **Setup** FE.Setup($1^k, n$) is a PPT algorithm that takes as input the security parameter k and the function arity n . It outputs n encryption keys $\text{EK}_1, \dots, \text{EK}_n$ and a master secret key MSK.
- **Encryption** FE.Enc(EK, x) is a PPT algorithm that takes as input an encryption key $\text{EK}_i \in (\text{EK}_1, \dots, \text{EK}_n)$ and an input message $x \in \mathcal{X}_k$ and outputs a ciphertext CT. In the case where all of the encryption keys EK_i are the same, we assume that each ciphertext CT has an associated label i to denote that the encrypted plaintext constitutes an i 'th input to a function $f \in \mathcal{F}_k$. For convenience of notation, we omit the labels from the explicit description of the ciphertexts. In particular, note that when EK_i 's are *distinct*, the index of the encryption key EK_i used to compute CT implicitly denotes that the plaintext encrypted in CT constitutes an i 'th input to f , and thus no explicit label is necessary.
- **Key Generation** FE.Keygen(MSK, f) is a PPT algorithm that takes as input the master secret key MSK and an n -ary function $f \in \mathcal{F}_k$ and outputs a corresponding secret key SK_f .
- **Decryption** FE.Dec($\text{SK}_f, \text{CT}_1, \dots, \text{CT}_n$) is a deterministic algorithm that takes as input a secret key SK_f and n ciphertexts $\text{CT}_1, \dots, \text{CT}_n$ and outputs a string $y \in \mathcal{Y}_k$.

Definition 1 (Correctness). *A multi-input functional encryption scheme \mathcal{FE} for \mathcal{F} is correct if for all $f \in \mathcal{F}_k$ and all $(x_1, \dots, x_n) \in \mathcal{X}_k^n$:*

$$\Pr \left[\begin{array}{l} (\vec{\text{EK}}, \text{MSK}) \leftarrow \text{FE.Setup}(1^k); \text{SK}_f \leftarrow \text{FE.Keygen}(\text{MSK}, f); \\ \text{FE.Dec}(\text{SK}_f, \text{FE.Enc}(\text{EK}_1, x_1), \dots, \text{FE.Enc}(\text{EK}_n, x_n)) \neq f(x_1, \dots, x_n) \end{array} \right] = \text{negl}(k)$$

where the probability is taken over the coins of FE.Setup, FE.Keygen and FE.Enc.

2.2 Security for Multi-Input Functional Encryption

We now present our security definitions for multi-input functional encryption. Following the literature on single-input FE, we consider two notions of security, namely, indistinguishability-based security (or IND-security, in short) and simulation-based security (or SIM-security, in short).

Notation. We start by introducing some notation that is used in our security definitions. Let \mathbb{N} denote the set of positive integers $\{1, \dots, n\}$ where n denotes the arity of functions. For any two sets $S = \{s_0, \dots, s_{|S|}\}$ and $I = \{i_1, \dots, i_{|I|}\}$ such that $|I| \leq |S|$, we let S_I denote the subset $\{s_i\}_{i \in I}$ of the set S . Throughout the text, we use the vector and set notation interchangeably, as per convenience. For simplicity of notation, we omit explicit reference to auxiliary input to the adversary from our definitions.

2.2.1 Indistinguishability-based Security

Here we present an indistinguishability-based security definition for multi-input FE.

Intuition. We start by giving an overview of the main ideas behind our indistinguishability-based security definition. To convey the core ideas, it suffices to consider the case of 2-ary functions. We will assume familiarity with the security definitions for single-input FE.

Let us start by considering the natural extension of *public-key* single-input FE to the two-input setting. That is, suppose there are two public encryption keys EK_1, EK_2 that are used to create ciphertexts of first inputs and second inputs, respectively, for 2-ary functions. Let us investigate what security can be achieved for *one* pair of challenge message tuples $(x_1^0, x_2^0), (x_1^1, x_2^1)$ for the simplified case where the adversary makes secret key queries after receiving the challenge ciphertexts.

Suppose that the adversary queries secret keys for functions $\{f\}$. Now, recall that the IND-security definition in the single-input case guarantees that an adversary cannot differentiate between encryptions of x^0 and x^1 as long as $f(x^0) = f(x^1)$ for every $f \in \{f\}$. We note, however, that an analogous security guarantee cannot be achieved in the multi-input setting. That is, restricting the functions $\{f\}$ to be such that $f(x_1^0, x_2^0) = f(x_1^1, x_2^1)$ is *not* enough since an adversary who knows both the encryption keys can create its own ciphertexts w.r.t. each encryption key. Then, by using the secret key corresponding to function f , it can learn additional values $\{f(x_1^b, \cdot)\}$ and $\{f(\cdot, x_2^b)\}$, where b is the challenge bit. In particular, if, for example, there exists an input x^* such that $f(x_1^0, x^*) \neq f(x_1^1, x^*)$, then the adversary can learn the challenge bit b ! Therefore, we must enforce additional restrictions on the query functions f . Specifically, we must require that $f(x_1^0, x') = f(x_1^1, x')$ for *every* input x' in the domain (and similarly $f(x', x_2^0) = f(x', x_2^1)$). Note that this restriction “grows” with the arity n of the functions.

Let us now consider the secret-key case, where all the encryption keys are secret. In this case, for the above example, it suffices to require that $f(x_1^0, x_2^0) = f(x_1^1, x_2^1)$ since the adversary cannot create its own ciphertexts. Observe, however, that when there are *multiple* challenge messages, then an adversary can learn function evaluations over different “combinations” of challenge messages. In particular, if there are q challenge messages per encryption key, then the adversary can learn q^2 output values for every f . Then, we must enforce that for every $i \in [q^2]$, the i ’th output value y_i^0 when challenge bit $b = 0$ is *equal* to the output value y_i^1 when the challenge bit $b = 1$.

The security guarantees in the public-key and the secret-key settings as discussed above are vastly different. In general, we observe that the *more* the number of encryption keys that are public, the *smaller* the class of functions that can be supported by the definition. Bellow, we present a unified definition that simultaneously captures the extreme cases of public-key and secret-key settings as well as all the “in between” cases.

Compatible Functions and Input Plaintexts. To facilitate the presentation of our IND security definition, we first introduce the following notion:

Definition 2 (I-Compatibility). *Let $\{f\}$ be any set of functions $f \in \mathcal{F}_k$. Let $\mathbf{N} = \{1, \dots, n\}$ and $\mathbf{I} \subseteq \mathbf{N}$. Let \vec{X}^0 and \vec{X}^1 be a pair of input vectors, where $\vec{X}^b = \{x_{1,j}^b, \dots, x_{n,j}^b\}_{j=1}^q$. We say that \mathcal{F} and (\vec{X}^0, \vec{X}^1) are I-compatible if they satisfy the following property:*

- *For every $f \in \{f\}$, every $\mathbf{I}' = \{i_1, \dots, i_t\} \subseteq \mathbf{I} \cup \emptyset$, every $j_1, \dots, j_{n-t} \in [q]$, and every $x'_{i_1}, \dots, x'_{i_t} \in \mathcal{X}_k$,*

$$f\left(\langle x_{i_1, j_1}^0, \dots, x_{i_{n-t}, j_{n-t}}^0, x'_{i_1}, \dots, x'_{i_t} \rangle\right) = f\left(\langle x_{i_1, j_1}^1, \dots, x_{i_{n-t}, j_{n-t}}^1, x'_{i_1}, \dots, x'_{i_t} \rangle\right),$$

where $\langle y_{i_1}, \dots, y_{i_n} \rangle$ denotes a permutation of the values y_{i_1}, \dots, y_{i_n} such that the value y_{i_j} is mapped to the ℓ ’th location if y_{i_j} is the ℓ ’th input (out of n inputs) to f .

IND-secure MI-FE. Our security definition is parameterized by two variables t and q , where t denotes the number of encryption keys known to the adversary, and q denotes the number of challenge messages per encryption key. Thus, in total, the adversary is allowed to make $Q = q \cdot n$ number of challenge message queries. We are now ready to present our formal definition for (t, q) -IND-secure multi-input functional encryption.

Definition 3 (Indistinguishability-based security). *We say that a multi-input functional encryption scheme \mathcal{FE} for n -ary functions \mathcal{F} is (t, q) -IND-secure if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, the advantage of \mathcal{A} defined as*

$$\text{Adv}_{\mathcal{A}}^{\mathcal{FE}, \text{IND}}(1^k) = \left| \Pr[\text{IND}_{\mathcal{A}}^{\mathcal{FE}}(1^k) = 1] - \frac{1}{2} \right|$$

is $\text{negl}(k)$, where:

Experiment $\text{IND}_{\mathcal{A}}^{\mathcal{FE}}(1^k)$:
 $(I, \text{st}_0) \leftarrow \mathcal{A}_0(1^k)$ where $|I| = t$
 $(\vec{E}\mathbf{K}, \text{MSK}) \leftarrow \text{FE.Setup}(1^k)$
 $(\vec{X}^0, \vec{X}^1, \text{st}_1) \leftarrow \mathcal{A}_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{st}_0, \vec{E}\mathbf{K}_I)$ where $\vec{X}^\ell = \{x_{1,j}^\ell, \dots, x_{n,j}^\ell\}_{j=1}^q$
 $b \leftarrow \{0, 1\}$; $\text{CT}_{i,j} \leftarrow \text{FE.Enc}(\vec{E}\mathbf{K}_i, x_{i,j}^b) \forall i \in [n], j \in [q]$
 $b' \leftarrow \mathcal{A}_2^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{st}_1, \vec{\text{CT}})$
Output: $(b = b')$

In the above experiment, we require:

- Let $\{f\}$ denote the entire set of key queries made by \mathcal{A}_1 . Then, the challenge message vectors \vec{X}^0 and \vec{X}^1 chosen by \mathcal{A}_1 must be I-compatible with $\{f\}$.
- The key queries $\{g\}$ made by \mathcal{A}_2 must be I-compatible with \vec{X}^0 and \vec{X}^1 .

Selective Security. We also consider selective *indistinguishability*-based security for multi-input functional encryption. Formally, (t, q) -sel-IND-security is defined in the same manner as Definition 3, except that the adversary \mathcal{A}_1 is required to choose the challenge message vectors \vec{X}^0, \vec{X}^1 before the evaluation keys $\vec{E}\mathbf{K}$ and the master secret key MSK are chosen by the challenger. We omit the formal definition to avoid repetition.

2.2.2 Simulation-based Security

Here we present a simulation-based security definition for multi-input FE. We consider the case where the adversary makes key queries *after* choosing the challenge messages. That is, we only consider *adaptive* key queries. The “opposite” case where the adversary makes key queries *before* choosing the challenge messages (i.e., *non-adaptive* key queries) is discussed in Section D.

Our definition extends the simulation-based security definition for single-input FE that supports adaptive key queries [BSW11, O’N10, BO13, CIJ⁺13]. In particular, we present a general definition that models both black-box and non-black-box simulation.

Intuition. We start by giving an overview of the main ideas behind our simulation-based security definition. To convey the core ideas, it suffices to consider the case of 2-ary functions. Let us start by considering the natural extension of *public-key* single-input FE to the two-input setting. That is, suppose there are two public encryption keys EK_1, EK_2 that are used to create ciphertexts of first inputs and second inputs, respectively, for 2-ary functions. Let us investigate what security can be achieved for *one* challenge message tuple (x_1, x_2) .

Suppose that the adversary queries secret keys for functions $\{f\}$. Now, recall that the SIM-security definition in the single-input case guarantees that for every $f \in \{f\}$, an adversary cannot learn more than $f(x)$ when x is the challenge message. We note, however, that an analogous security guarantee cannot be achieved in the multi-input setting. Indeed, an adversary who knows both the encryption keys can create its own ciphertexts w.r.t. each encryption key. Then, by using the secret key corresponding to function f , it can learn additional values $\{f(x_1, \cdot)\}$ and $\{f(\cdot, x_2)\}$. Thus, we must allow for the ideal world adversary, aka simulator, to learn the same information.

In the secret-key case, however, since all of the encryption keys are secret, the SIM-security definition for single-input FE indeed extends in a natural manner to the multi-input setting. We stress, however, that when there are multiple challenge messages, we must take into account the fact that adversary can learn function evaluations over all possible “combinations” of challenge messages. Our definition presented below formalizes this intuition.

SIM-secure MI-FE. Similar to the IND-security case, our definition is parameterized by variables t and q as defined earlier. We now formally define (t, q) -SIM-secure multi-input functional encryption.

Definition 4 (Simulation-based Security). *We say that a functional encryption scheme \mathcal{FE} for n -ary functions \mathcal{F} is (t, q) -SIM-secure if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ such that the outputs of the following two experiments are computationally indistinguishable:*

Experiment $\text{REAL}_{\mathcal{A}}^{\mathcal{FE}}(1^k)$: $(I, \text{st}_0) \leftarrow \mathcal{A}_0(1^k)$ where $ I = t$ $(\vec{E}\vec{K}, \text{MSK}) \leftarrow \text{FE.Setup}(1^k)$ $(\mathcal{M}, \text{st}_1) \leftarrow \mathcal{A}_1(\text{st}_0, \vec{E}\vec{K}_I)$ $\vec{X} \leftarrow \mathcal{M}$ where $\vec{X} = \{x_{1,j}, \dots, x_{n,j}\}_{j=1}^q$ $\text{CT}_{i,j} \leftarrow \text{FE.Enc}(\text{EK}_i, x_{i,j}) \forall i \in [n], j \in [q]$ $\alpha \leftarrow \mathcal{A}_2^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\vec{\text{CT}}, \text{st}_1)$ Output: $(I, \mathcal{M}, \vec{X}, \{f\}, \alpha)$	Experiment $\text{IDEAL}_{\mathcal{S}}^{\mathcal{FE}}(1^k)$: $(I, \text{st}_0) \leftarrow \mathcal{S}_0(1^k)$ $(\mathcal{M}, \text{st}_1) \leftarrow \mathcal{S}_1(\text{st}_0)$ $\alpha \leftarrow \mathcal{S}_2^{\text{TP}(\mathcal{M}, \cdot, \cdot)}(\text{st}_1)$ Output: $(I, \mathcal{M}, \vec{X}, \{g\}, \alpha)$
--	--

where the oracle $\text{TP}(\mathcal{M}, \cdot, \cdot)$ denotes the ideal world trusted party, $\{f\}$ denotes the set of queries of \mathcal{A}_2 to FE.Keygen and $\{g\}$ denotes the set of functions appearing in the queries of \mathcal{S}_2 to TP. Given the message distribution \mathcal{M} , TP first samples a message vector $\vec{X} \leftarrow \mathcal{M}$, where $\vec{X} = \{x_{1,j}, \dots, x_{n,j}\}_{j=1}^q$. It then accepts queries of the form $(g, (j_1, \dots, j_{n-p}), (x'_{i'_1}, \dots, x'_{i'_p}))$ where $p \leq t$, $\{i'_1, \dots, i'_p\} \subseteq I \cup \emptyset$ and $x'_{i'_1}, \dots, x'_{i'_p} \in \mathcal{X}_k$. On receiving such a query, TP outputs:

$$g\left(\left\langle x_{i_1, j_1}, \dots, x_{i_{n-p}, j_{n-p}}, x'_{i'_1}, \dots, x'_{i'_p} \right\rangle\right),$$

where $\langle y_{i_1}, \dots, y_{i_n} \rangle$ denotes a permutation of the values y_{i_1}, \dots, y_{i_n} such that the value y_{i_j} is mapped to the ℓ 'th location if y_{i_j} is the ℓ 'th input (out of n inputs) to g .

Remark 5 (On Queries to the Trusted Party). Note that when $t = 0$, then given the challenge ciphertexts $\vec{\text{CT}}$, intuitively, the real adversary can only compute values $\text{FE.Dec}(\text{SK}_f, \text{CT}_{1, j_1}, \dots, \text{CT}_{n, j_n})$ for every $j_i \in [q]$, $i \in [n]$. To formalize the intuition that this adversary does not learn anything more than function values $\{f(x_{1, j_1}, \dots, x_{n, j_n})\}$, we restrict the ideal adversary aka simulator to learn exactly this information.

However, when $t > 0$, then the real adversary can compute values:

$$\text{FE.Dec} \left(\text{SK}_f, \left\langle \text{CT}_{i_1, j_1}, \dots, \text{CT}_{i_{n-t}, j_{n-t}}, \text{CT}'_{i'_1}, \dots, \text{CT}'_{i'_t} \right\rangle \right)$$

for ciphertexts $\text{CT}'_{i'_t}$ of its choice since it knows the encryption keys $\vec{\text{EK}}_I$. In other words, such an adversary can learn function values of the form $f(\langle x_{i_1, j_1}, \dots, x_{i_{n-t}, j_{n-t}}, \cdot, \dots, \cdot \rangle)$. Thus, we must provide the same ability to the simulator as well. Our definition presented above precisely captures this.

Selective Security. We also consider *selective* simulation-based security for multi-input functional encryption. Formally, (t, q) -sel-SIM-security is defined in the same manner as Definition 4, except that in the real world experiment, adversary \mathcal{A}_1 chooses the message distribution \mathcal{M} before the evaluation keys $\vec{\text{EK}}$ and the master secret key MSK are chosen by the challenger. We omit the formal definition to avoid repetition.

Remark 6 (SIM-security: Secret-key setting). When $t = 0$, none of the encryption keys are known to the adversary. In this “secret-key” setting, there is no difference between $(0, q)$ -sel-SIM-security and $(0, q)$ -SIM-security.

3 Preliminaries

Here we present definitions of various cryptographic primitives that are used in our construction of multi-input functional encryption. We assume familiarity with standard semantically-secure public-key encryption and omit its formal definition from this text. Below, we recall the notions of indistinguishability obfuscation, non-interactive witness indistinguishable proof systems and perfectly binding commitment schemes.

3.1 Indistinguishability Obfuscation

Here we recall the notion of indistinguishability obfuscation that was defined by Barak et al. [BGI⁺01]. Intuitively speaking, we require that for any two circuits C_1 and C_2 that are “functionally equivalent” (i.e., for all inputs x in the domain, $C_1(x) = C_2(x)$), the obfuscation of C_1 must be computationally indistinguishable from the obfuscation of C_2 . Below we present the formal definition following the syntax of [GGH⁺13a].

Definition 7 (Indistinguishability Obfuscation). A uniform PPT machine $i\mathcal{O}$ is called an *indistinguishability obfuscator* for a circuit class $\{\mathcal{C}_k\}$ if the following holds:

- **Correctness:** For every $k \in \mathbb{N}$, for every $C \in \mathcal{C}_k$, for every input x in the domain of C , we have that

$$\Pr[C'(x) = C(x) : C' \leftarrow i\mathcal{O}(C)] = 1.$$

- **Indistinguishability:** For every $k \in \mathbb{N}$, for all pairs of circuits $C_0, C_1 \in \mathcal{C}_k$, if $C_0(x) = C_1(x)$ for all inputs x , then for all PPT adversaries \mathcal{A} , we have:

$$|\Pr[\mathcal{A}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{A}(i\mathcal{O}(C_1)) = 1]| \leq \text{negl}(k).$$

Very recently, Garg et al. [GGH⁺13a] gave the first candidate construction for an indistinguishability obfuscator $i\mathcal{O}$ for the circuit class P/poly . ADD.

Differing-Inputs Obfuscation. We also consider a stronger notion of indistinguishability obfuscation, namely, *differing-inputs obfuscation* that was proposed by Barak et al [BGI⁺01]. Intuitively speaking, we require that for any two circuits C_1 and C_2 that “appear” to be functionally equivalent to every PPT algorithm (i.e., no PPT algorithm can find an input x s.t. $C_1(x) \neq C_2(x)$), the obfuscation of C_1 must be computationally indistinguishable from the obfuscation of C_2 . Alternatively, if a PPT algorithm can distinguish obfuscation of C_1 from obfuscation of C_2 , then we can *efficiently* find an input x s.t. $C_1(x) \neq C_2(x)$.

Below, we present the formal definition. We follow the formalism of [ABG⁺13].

We start by defining the notion of differing-inputs circuit family. Intuitively, a circuit family is said to be a differing-inputs circuits family if there does not exist any PPT adversary that given two circuits, that are sampled from a distribution defined on this circuit family, can find an input x such that both the circuits yield different outputs on x .

Definition 8 (Differing-Inputs Circuit Family). *A circuit family \mathcal{C} associated with a sampler Sampler is said to be a differing-inputs circuit family if for every PPT adversary \mathcal{A} , there exists a negligible function $\text{negl}(\cdot)$ such that:*

$$\Pr [C_0(x) \neq C_1(x) \mid (C_0, C_1, z) \leftarrow \text{Sampler}(1^k); x \leftarrow \mathcal{A}(1^k, C_0, C_1, z)] \leq \text{negl}(k)$$

Definition 9 (Differing-Inputs Obfuscator). *A PPT machine diO is called a differing-inputs obfuscator for a differing-inputs circuits family $\mathcal{C} = \{C_k\}$ if the following conditions are satisfied:*

- **Correctness:** *For all security parameters $k \in \mathbb{N}$, for all $C \in \mathcal{C}$, for all inputs x , we have that:*

$$\Pr[C'(x) = C(x) \mid C' \leftarrow \text{diO}(1^k, C)] = 1$$

- **Differing-inputs:** *For any PPT adversary \mathcal{A} , there exists a negligible function $\text{negl}(\cdot)$ such that the following holds: For all security parameters $k \in \mathbb{N}$, for $(C_0, C_1, z) \leftarrow \text{Sampler}(1^k)$, we have that:*

$$|\Pr[\mathcal{A}(\text{diO}(C_1)) = 1] - \Pr[\mathcal{A}(\text{diO}(C_2)) = 1]| \leq \text{negl}(k).$$

3.2 Non-Interactive Proof Systems

In this section, we recall various security notions for non-interactive proof systems. We start by giving the syntax and formal definition of a non-interactive proof system. Next, we give the definition of non-interactive witness-indistinguishable proofs (NIWI). Finally, we give the definition of non-interactive zero-knowledge (NIZK), with simulation-soundness property.

Syntax. Let R be an efficiently computable relation that consists of pairs (x, w) , where x is called the statement and w is the witness. Let L denote the language consisting of statements in R . A non-interactive proof system for a language L consists of a setup algorithm CRSGen , a prover algorithm Prove and a verifier algorithm Verify , defined as follows:

- **Setup** $\text{CRSGen}(1^k)$ is a PPT algorithm that takes as input the security parameter k and outputs a common reference string crs .
- **Prover** $\text{Prove}(\text{crs}, x, w)$ is a PPT algorithm that takes as input the common reference string CRS , a statement x along with a witness w . $(x, w) \in R$; if so, it produces a proof string π , else it outputs **fail**.
- **Verifier** $\text{Verify}(\text{crs}, x, \pi)$ is a PPT algorithm that takes as input the common reference string crs and a statement x with a corresponding proof π . It outputs 1 if the proof is valid, and 0 otherwise.

Definition 10 (Non-interactive Proof System). A non-interactive proof system for a language L with a PPT relation R is a tuple of algorithms $(\text{CRSGen}, \text{Prove}, \text{Verify})$ such that the following properties hold:

- **Perfect Completeness:** For every $(x, w) \in R$, it holds that

$$\Pr[\text{Verify}(\text{crs}, x, \text{Prove}(\text{crs}, x, w)) = 1] = 1$$

where $\text{crs} \leftarrow \text{CRSGen}(1^k)$, and the probability is taken over the coins of CRSGen , Prove and Verify .

- **Statistical Soundness:** For every adversary \mathcal{A} , it holds that

$$\Pr[\text{Verify}(\text{crs}, x, \pi) = 1 \wedge x \notin L \mid \text{crs} \leftarrow \text{CRSGen}(1^k); (x, \pi) \leftarrow \mathcal{A}(\text{crs})] = \text{negl}(k)$$

If the soundness property only holds against PPT adversaries, then we call it an argument system.

Definition 11 (NIWI). We say that a non-interactive proof system $(\text{CRSGen}, \text{Prove}, \text{Verify})$ for a language L with a PPT relation R is witness-indistinguishable if for any triplet (x, w_0, w_1) such that $(x, w_0) \in R$ and $(x, w_1) \in R$, the distributions $\{\text{crs}, \text{Prove}(\text{crs}, x, w_0)\}$ and $\{\text{crs}, \text{Prove}(\text{crs}, x, w_1)\}$ are computationally indistinguishable, where $\text{crs} \leftarrow \text{CRSGen}(1^k)$.

Definition 12 (NIZK). A non-interactive proof system $(\text{CRSGen}, \text{Prove}, \text{Verify})$ for a language L with a PPT relation R is said to be zero knowledge if there exists a simulator $\text{Sim} = (\text{Sim.CRSGen}, \text{Sim.Prove})$ such that for all PPT adversaries \mathcal{A} ,

$$\left| \Pr[\mathcal{A}^{\text{Prove}(\text{crs}, \cdot, \cdot)}(\text{crs}) = 1 \mid \text{crs} \leftarrow \text{CRSGen}(1^k)] - \Pr[\mathcal{A}^{\mathcal{S}(\text{crs}, \tau, \cdot, \cdot)}(\text{crs}) = 1 \mid (\text{crs}, \tau) \leftarrow \text{Sim.CRSGen}(1^k)] \right| = \text{negl}(k)$$

where $\mathcal{S}(\text{crs}, \tau, x, w) = \text{Sim.Prove}(\text{crs}, \tau, x)$ if $(x, w) \in R$ and outputs fail otherwise.

Definition 13 (Simulation soundness). A NIZK proof system $(\text{CRSGen}, \text{Prove}, \text{Verify})$ for a language L with a PPT relation R is said to be simulation sound if for all PPT adversaries,

$$\Pr \left[\begin{array}{l} (x^*, \pi^*) \leftarrow \mathcal{A}^{\text{Sim.Prove}(\text{crs}, \tau, \cdot)}(\text{crs}) \wedge x^* \notin L \\ \wedge 1 \leftarrow \text{Verify}(\text{crs}, x^*, \pi^*) \mid (\text{crs}, \tau) \leftarrow \text{Sim.CRSGen}(1^k) \end{array} \right] = \text{negl}(k)$$

where x^* is not in the list of queries made by \mathcal{A} to Sim.Prove .

3.3 Commitment Schemes

A commitment scheme Com is a PPT algorithm that takes as input a string x and randomness r and outputs $c \leftarrow \text{Com}(x; r)$. A perfectly binding commitment scheme must satisfy the following properties:

- **Perfectly Binding:** This property states that two different strings cannot have the same commitment. More formally, $\forall x_1 \neq x_2$ and r_1, r_2 , $\text{Com}(x_1; r_1) \neq \text{Com}(x_2; r_2)$.
- **Computational Hiding:** For all strings x_0 and x_1 (of the same length), for all PPT adversaries \mathcal{A} , we have that:

$$|\Pr[A_1(\text{Com}(x_0)) = 1] - \Pr[A_1(\text{Com}(x_1)) = 1]| \leq \text{negl}(k).$$

We note that it is in fact sufficient to use a standard 2-round statistically binding scheme in our construction in Section 4. Note that such a commitment scheme can be based on one way functions. For simplicity of exposition, however, we will present our construction using a non-interactive perfectly binding scheme.

4 A Construction from Indistinguishability Obfuscation

Let \mathcal{F} denote the family of all efficiently computable (deterministic) n -ary functions. We now present a functional encryption scheme \mathcal{FE}_1 for \mathcal{F} . Assuming the existence of one-way functions and indistinguishability obfuscation for all efficiently computable circuits, we prove the following security guarantees for \mathcal{FE}_1 :

1. For $t = 0$, and any $q = q(k)$ such that $\binom{qn}{n} = \text{poly}(k)$, \mathcal{FE}_1 is $(0, q)$ -SIM-secure.² In this case, the size of the secret keys in \mathcal{FE}_1 grows linearly with $\binom{qn}{n}$.
2. For any $t \leq n$ and $q = \text{poly}(k)$, \mathcal{FE}_1 is (t, q) -sel-IND-secure. In this case, the size of the secret keys is independent of q .

Further, the size of each encryption key and ciphertext in \mathcal{FE}_1 grows linearly with q . In Section 5, we give an efficient construction with “compact” encryption keys and ciphertexts, whose security is proven in the standard model.

Notation. Let $(\text{CRSGen}, \text{Prove}, \text{Verify})$ be a NIWI proof system. Let Com denote a perfectly binding commitment scheme. Let $i\mathcal{O}$ denote an indistinguishability obfuscator. Finally, let $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ be a semantically secure public-key encryption scheme. (See Section 3 for definitions of these notions.) We denote the length of ciphertexts in PKE by $\text{c-len} = \text{c-len}(k)$. Let $\text{len} = 2 \cdot \text{c-len}$.

We now proceed to describe our scheme $\mathcal{FE}_1 = (\text{FE.Setup}, \text{FE.Enc}, \text{FE.Keygen}, \text{FE.Dec})$.

Setup $\text{FE.Setup}(1^k)$: The setup algorithm first computes a CRS $\text{crs} \leftarrow \text{CRSGen}(1^k)$ for the NIWI proof system. Next, it computes two key pairs $(\text{pk}_1, \text{sk}_1) \leftarrow \text{PKE.Setup}(1^k)$ and $(\text{pk}_2, \text{sk}_2) \leftarrow \text{PKE.Setup}(1^k)$ of the public-key encryption scheme PKE . Finally, it computes the following commitments: (a) $Z_1^{i,j} \leftarrow \text{Com}(0^{\text{len}})$ for every $i \in [n]$, $j \in [q]$. (b) $Z_2^i \leftarrow \text{Com}(0)$ for every $i \in [n]$.

For every $i \in [n]$, the i 'th encryption key $\text{EK}_i = (\text{crs}, \text{pk}_1, \text{pk}_2, \{Z_1^{i,j}\}, Z_2^i, r_2^i)$ where r_2^i is the randomness used to compute the commitment Z_2^i . The master secret key is set to be $\text{MSK} = (\text{crs}, \text{pk}_1, \text{pk}_2, \text{sk}_1, \{Z_1^{i,j}\}, \{Z_2^i\})$. The setup algorithm outputs $(\text{EK}_1, \dots, \text{EK}_n, \text{MSK})$.

Encryption $\text{FE.Enc}(\text{EK}_i, x)$: To encrypt a message x with the i 'th encryption key EK_i , the encryption algorithm first computes $c_1 \leftarrow \text{PKE.Enc}(\text{pk}_1, x)$ and $c_2 \leftarrow \text{PKE.Enc}(\text{pk}_2, x)$. Next, it computes a NIWI proof $\pi \leftarrow \text{Prove}(\text{crs}, y, w)$ for the statement $y = (c_1, c_2, \text{pk}_1, \text{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$:

- Either c_1 and c_2 are encryptions of the same message and Z_2^i is a commitment to 0, or
- $\exists j \in [q]$ s.t. $Z_1^{i,j}$ is a commitment to $c_1 \| c_2$.

A witness $w_{\text{real}} = (m, s_1, s_2, r_2^i)$ for the first part of the statement, referred to as the *real witness*, includes the message m and the randomness s_1 and s_2 used to compute the ciphertexts c_1 and c_2 , respectively, and the randomness r_2^i used to compute Z_2^i . A witness $w_{\text{trap}} = (j, r_1^{i,j})$ for the second part of the statement, referred to as the *trapdoor witness*, includes an index j and the randomness $r_1^{i,j}$ used to compute $Z_1^{i,j}$.

The honest encryption algorithm uses the real witness w_{real} to compute π . The output of the algorithm is the ciphertext $\text{CT} = (c_1, c_2, \pi)$.

²Recall that when $t = 0$, there is no difference between selective security and standard security as defined in Section 2.2.2. See Remark 6.

Key Generation $\text{FE.Keygen}(\text{MSK}, f)$: The key generation algorithm on input f computes $\text{SK}_f \leftarrow i\mathcal{O}(\text{G}_f)$ where the function G_f is defined in Figure 1. Note that G_f has the master secret key MSK hardwired in its description.

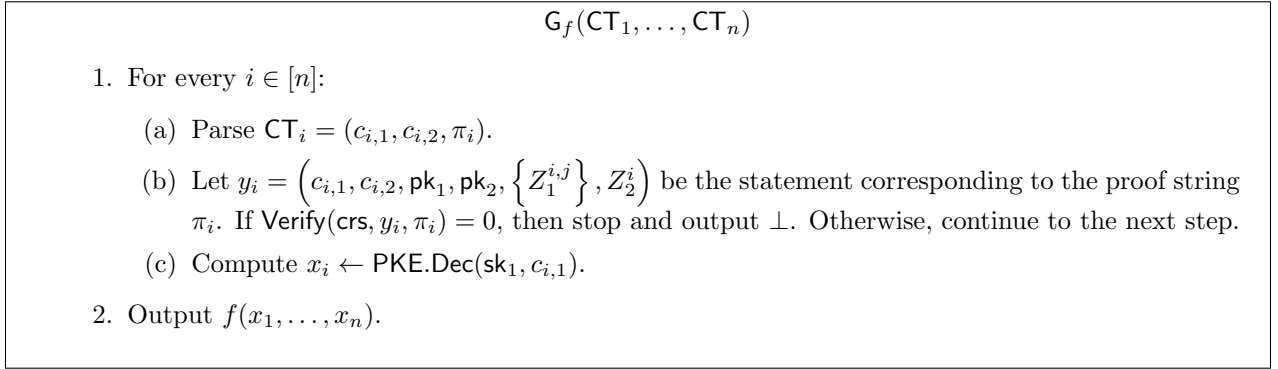


Figure 1: Functionality G_f

The algorithm outputs SK_f as the secret key for f .

Size of Function G_f . In order to prove that \mathcal{FE}_1 is $(0, q)$ -SIM-secure (see Section 4.2), we require the function G_f to be padded with zeros such that $|\text{G}_f| = |\text{Sim.G}_f|$, where the “simulated” functionality Sim.G_f is described later in Figure 2. In this case, the size of SK_f grows linearly with $\binom{qn}{n}$.

Note, however, that such a padding is *not* necessary to prove (t, q) -sel-IND-security for \mathcal{FE}_1 (see Section 4.1). Indeed, in this case, the secret keys SK_f are independent of the number of message queries q made by the adversary.

Decryption $\text{FE.Dec}(\text{SK}_f, \text{CT}_1, \dots, \text{CT}_n)$: The decryption algorithm on input $(\text{CT}_1, \dots, \text{CT}_n)$ computes and outputs $\text{SK}_f(\text{CT}_1, \dots, \text{CT}_n)$.

This completes the description of the proposed functional encryption scheme \mathcal{FE}_1 . The correctness property of the scheme follows from inspection. We prove sel-IND security for \mathcal{FE}_1 in Section 4.1, and then prove SIM security in Section 4.2.

4.1 Proving sel-IND Security

We now prove that the proposed scheme \mathcal{FE}_1 is (t, q) -sel-IND-secure for any $t \leq n$.

Theorem 14. *Let $q = q(k)$ be a fixed $\text{poly}(k)$. Then, assuming indistinguishability obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme \mathcal{FE}_1 is (t, q) -sel-IND-secure for any $t \leq n$.*

We prove the above theorem via a hybrid argument. We start by describing a sequence of hybrid experiments $\text{H}_0, \dots, \text{H}_{10}$, where experiment H_0 (resp., H_{10}) corresponds to the real world experiment with challenge bit $b = 0$ (resp., $b = 1$). We will prove that for every i , the outputs of experiments H_i and H_{i+1} are computationally indistinguishable.

Hybrid H_0 : This is the real experiment with challenge bit $b = 0$.

Hybrid H_1 : This experiment is the same as H_0 except that the setup algorithm computes the commitments $\{Z_1^{i,j}\}$ in the following manner: let the challenge ciphertext $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Then, $Z_1^{i,j} \leftarrow \text{Com}(\hat{c}_1^{i,j} \parallel \hat{c}_2^{i,j})$.

Hybrid H₂: This experiment is the same as H₁ except that in every challenge ciphertext $\hat{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the proof string $\hat{\pi}^{i,j}$ is computed using the *trapdoor* witness.

Hybrid H₃: This experiment is the same as H₂ except that for every $i \in \mathbb{N} \setminus I$ (where I denotes the set of indices i s.t. EK_i is known to the adversary) the setup algorithm computes Z_2^i as a commitment to 1 (instead of 0). That is, for every $i \in [n]$, $Z_2^i \leftarrow \text{Com}(1)$.

Hybrid H₄: This experiment is the same as H₃ except that in every challenge ciphertext $\hat{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *second* ciphertext $\hat{c}_2^{i,j}$ is computed as an encryption of the challenge message $x_{i,j}^1$ (as opposed to $x_{i,j}^0$), i.e., $\hat{c}_2^{i,j} \leftarrow \text{FE.Enc}(EK_i, x_{i,j}^1)$.

Hybrid H₅: This experiment is the same as H₄ except that for every key query f , the corresponding secret key SK_f is computed as $SK_f \leftarrow i\mathcal{O}(G'_f)$ where G'_f is the same as the function G_f except that:

1. It has secret key sk_2 hardwired instead of sk_1 .
2. It decrypts the *second* component of each input ciphertext using sk_2 . More concretely, in step 1(c), plaintext x'_i is computed as $x'_i \leftarrow \text{PKE.Dec}(sk_2, c_{i,2})$.

Hybrid H₆: This experiment is the same as H₅ except that in every challenge ciphertext $\hat{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *first* ciphertext $\hat{c}_1^{i,j}$ is an encryption of challenge message $x_{i,j}^1$ (as opposed to $x_{i,j}^0$), i.e., $\hat{c}_1^{i,j} \leftarrow \text{FE.Enc}(EK_i, x_{i,j}^1)$.

Hybrid H₇: This experiment is the same as H₆ except that for every key query f , the corresponding secret key SK_f is computed as $SK_f \leftarrow i\mathcal{O}(G_f)$.

Hybrid H₈: This experiment is the same as H₇ except that the setup algorithm computes every Z_2^i as a commitment to 0, i.e., $Z_2^i \leftarrow \text{Com}(0)$.

Hybrid H₉: This experiment is the same as H₈ except that in every challenge ciphertext $\hat{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the proof string $\hat{\pi}^{i,j}$ is computed using the *real* witness.

Hybrid H₁₀: This experiment is the same as H₉ except that the setup algorithm computes every $Z_1^{i,j}$ as a commitment to the all zeros string, i.e., $Z_1^{i,j} \leftarrow \text{Com}(0^{\text{len}})$. Note that this is the real experiment with challenge bit $b = 1$.

This completes the description of the hybrids. We argue their indistinguishability in Appendix A.

4.2 Proving SIM Security

Here we prove that the proposed scheme \mathcal{FE}_1 is $(0, q)$ -SIM-secure.

Theorem 15. *Let $q = q(k)$ be such that $\binom{qn}{n} = \text{poly}(k)$. Then, assuming indistinguishability obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme \mathcal{FE}_1 is $(0, q)$ -SIM-secure.*

In order to prove the above theorem, we first construct an ideal world adversary aka simulator \mathcal{S} . Then, in Appendix B, we prove indistinguishability of the outputs of the real and ideal world experiments via a hybrid argument.

Simulator \mathcal{S} . We describe a simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ that only makes black-box use of a real-world adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$.

ALGORITHM \mathcal{S}_0 . Let z be the auxiliary input given to \mathcal{S} . Algorithm \mathcal{S}_0 simply runs \mathcal{A}_0 with auxiliary input z and outputs $(I, \text{st}_0) \leftarrow \mathcal{A}_0(1^k, z)$. Since we are only considering the case where $t = 0$, we have that $I = \emptyset$.

ALGORITHM \mathcal{S}_1 . Algorithm \mathcal{S}_1 simply runs \mathcal{A}_1 on input st_0 and outputs $(\mathcal{M}, \text{st}_1) \leftarrow \mathcal{A}_0(\text{st}_0)$.

ALGORITHM \mathcal{S}_2 . This algorithm runs the adversary algorithm \mathcal{A}_2 on simulated ciphertexts and provides simulated answers to the key queries made by \mathcal{A}_2 . More concretely, \mathcal{S}_2 runs in the following sequence of steps:

1. Simulate Setup. \mathcal{S}_2 first performs a simulated setup procedure. Namely, it first computes a CRS $\text{crs} \leftarrow \text{CRSGen}(1^k)$ for the NIWI proof system. Next, it computes two key pairs $(\text{pk}_1, \text{sk}_1) \leftarrow \text{PKE.Setup}(1^k)$ and $(\text{pk}_2, \text{sk}_2) \leftarrow \text{PKE.Setup}(1^k)$ of the public-key encryption scheme PKE. Finally, it computes the commitments $\{Z_1^{i,j}\}$ and $\{Z_2^i\}$ in the following manner:

- For every $i \in [n], j \in [q]$: (a) Compute $\hat{c}_1^{i,j}$ and $\hat{c}_2^{i,j}$ as encryptions of zeros, i.e., $\hat{c}_1^{i,j} \leftarrow \text{PKE.Enc}(\text{pk}_1, 0)$ and $\hat{c}_2^{i,j} \leftarrow \text{PKE.Enc}(\text{pk}_2, 0)$. (b) Compute $Z_1^{i,j} \leftarrow \text{Com}(\hat{c}_1^{i,j} \parallel \hat{c}_2^{i,j})$. Let $r_1^{i,j}$ be the randomness used to compute $Z_1^{i,j}$.
- For every $i \in [n]$, compute Z_2^i as a commitment to 1, i.e., $Z_2^i \leftarrow \text{Com}(1)$.

Let $\text{MSK} = (\text{crs}, \text{pk}_1, \text{pk}_2, \text{sk}_1, \{Z_1^{i,j}\}, \{Z_2^i\})$.

2. Simulate Challenge Ciphertexts. \mathcal{S}_2 now computes simulated challenge ciphertexts $\vec{\text{CT}} = \{\hat{\text{CT}}_{1,j}, \dots, \hat{\text{CT}}_{n,j}\}_{j=1}^q$ in the following manner. For every $i \in [n], j \in [q]$:

- Let $y_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \text{pk}_1, \text{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$. Compute the proof $\hat{\pi}_i \leftarrow \text{Prove}(\text{crs}, y_{i,j}, w_{i,j})$ where the witness $w_{i,j}$ corresponds to the *trapdoor witness* $(j, r_1^{i,j})$. That is, $w_{i,j}$ establishes that $Z_1^{i,j}$ is a commitment to $\hat{c}_1^{i,j} \parallel \hat{c}_2^{i,j}$.
- The simulated ciphertext $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}_i)$.

3. Simulate Key Queries. Finally, \mathcal{S}_2 runs the adversary algorithm \mathcal{A}_2 on input $(\vec{\text{CT}}, \text{st}_1)$. Recall from Definition 4 that \mathcal{A}_2 also makes queries to the key generation oracle. \mathcal{S}_2 simulates responses to \mathcal{A}_2 's key queries in the following manner. Let TP denote the ideal world trusted party that given the message distribution \mathcal{M} (output by \mathcal{S}_2) first samples $\vec{X} \leftarrow M$, where $\vec{X} = \{x_{1,j}, \dots, x_{n,j}\}_{j=1}^q$. When \mathcal{A}_2 makes a key query f , \mathcal{S}_2 performs the following sequence of steps:

- Query the trusted party TP on function (f, j_1, \dots, j_n) for every choice of $j_1, \dots, j_n \in [q]$. The trusted party computes and returns the function outputs $\text{out}[j_1, \dots, j_n] = f(x_{1,j_1}, \dots, x_{n,j_n})$ to \mathcal{S}_2 . Let $\vec{\text{out}}$ denote the vector of all the $\binom{q}{n}$ number of outputs.
- Compute the secret key SK_f for function f as $\text{SK}_f \leftarrow i\mathcal{O}(\text{Sim.G}_f)$. The functionality Sim.G_f has the master secret key MSK , the challenge ciphertext pairs $\{\hat{c}_1^{i,j}, \hat{c}_2^{i,j}\}$ and the outputs $\vec{\text{out}}$ hardwired in it. It is described in Figure 2.
- Return SK_f to \mathcal{A}_2 .

Finally, at some point \mathcal{A}_2 outputs its view α . \mathcal{S}_2 outputs α and stops.

In Appendix B, we prove indistinguishability of the outputs of the real and ideal experiments.

5 A Construction from Differing-Inputs Obfuscation

Let \mathcal{F} denote the family of all efficiently computable (deterministic) n -ary functions. We now present a new functional encryption scheme \mathcal{FE}_{Π} for \mathcal{F} based on differing-inputs obfuscation.

$\text{Sim.G}_f(\text{CT}_1, \dots, \text{CT}_n)$

1. For every $i \in [n]$:
 - Parse $\text{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$.
 - Let $y_i = (c_{i,1}, c_{i,2}, \text{pk}_1, \text{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$ be the statement corresponding to the proof string π_i . If $\text{Verify}(\text{crs}, y_i, \pi_i) = 0$, then stop and output \perp . Otherwise, continue to the next step.
2. If $\exists (j_1, \dots, j_n)$ s.t. for every $i \in [n]$,
 - $\hat{c}_1^{i,j_i} = c_{i,1}$, and
 - $\hat{c}_2^{i,j_i} = c_{i,2}$,
 then stop and output $\text{out}[j_1, \dots, j_n]$.
3. Otherwise, for every $i \in [n]$,
 - Compute $x_i \leftarrow \text{PKE.Dec}(\text{sk}_1, c_{i,1})$.
4. Output $f(x_1, \dots, x_n)$.

Figure 2: Functionality Sim.G_f

The main advantage of this scheme over the one presented in Section 4 is that the encryption keys and the ciphertexts are “compact”, i.e., independent of the number of message queries q made by the adversary.

The proposed scheme provides the following security guarantees:

- For any choice of $t \leq n$, \mathcal{FE}_{II} is $(t, \text{poly}(k))$ -IND-secure. In this case, the number of message queries q can be an arbitrary unbounded polynomial $q = \text{poly}(k)$.
- For $t = 0$ and $q = q(k)$ such that $\binom{qn}{n} = \text{poly}(k)$, our construction naturally extends to $(0, q)$ -SIM-security. In this case, the size of the secret keys grows linearly with $\binom{qn}{n}$.

Notation. Let $(\text{CRSGen}, \text{Prove}, \text{Verify})$ be a simulation-sound NIZK argument system. Let Com denote a perfectly binding commitment scheme. Let diO denote a differing-inputs obfuscator. Finally, let $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ be a semantically secure public-key encryption scheme.

We now proceed to describe the scheme $\mathcal{FE}_{\text{II}} = (\text{FE.Setup}, \text{FE.Enc}, \text{FE.Keygen}, \text{FE.Dec})$.

Setup $\text{FE.Setup}(1^k)$: The setup algorithm first computes a CRS $\text{crs} \leftarrow \text{CRSGen}(1^k)$ for the simulation-sound NIZK proof system. Next, it computes two key pairs $(\text{pk}_1, \text{sk}_1) \leftarrow \text{PKE.Setup}(1^k)$ and $(\text{pk}_2, \text{sk}_2) \leftarrow \text{PKE.Setup}(1^k)$ – of the public-key encryption scheme PKE . Finally, for every $i \in [n]$, it computes a commitment $Z_i \leftarrow \text{Com}(0)$.

For every $i \in [n]$, the i ’th encryption key $\text{EK}_i = (\text{crs}, \text{pk}_1, \text{pk}_2, Z_i, r_i)$ where r_i is the randomness used to compute Z_i . The master secret key $\text{MSK} = (\text{crs}, \text{pk}_1, \text{pk}_2, \text{sk}_1, \{Z_i\})$. The setup algorithm outputs $(\text{EK}_1, \dots, \text{EK}_n, \text{MSK})$.

Encryption $\text{FE.Enc}(\text{EK}_i, x)$: To encrypt a message x with the i ’th encryption key EK_i , the encryption algorithm first computes $c_1 \leftarrow \text{PKE.Enc}(\text{pk}_1, x)$ and $c_2 \leftarrow \text{PKE.Enc}(\text{pk}_2, x)$. Next, it computes a simulation-sound NIZK proof $\pi \leftarrow \text{Prove}(\text{crs}, y, w)$ for the statement $y = (c_1, c_2, \text{pk}_1, \text{pk}_2, Z)$:

- c_1 and c_2 are encryptions of the same message *and* Z_i is a commitment to 0.

Here, a witness $w = (s_1, s_2, r_i)$ for y consists of the randomness s_1 and s_2 used to compute c_1 and c_2 , respectively, and the randomness r_i used to compute Z_i .

The output of the algorithm is the ciphertext $\text{CT} = (c_1, c_2, \pi)$.

Key Generation $\text{FE.Keygen}(\text{MSK}, f)$: The key generation algorithm on input f computes $\text{SK}_f \leftarrow \text{diO}(\mathcal{H}_f)$ where the function \mathcal{H}_f is defined in Figure 3. Note that \mathcal{H}_f has the master secret key MSK hardwired in its description.

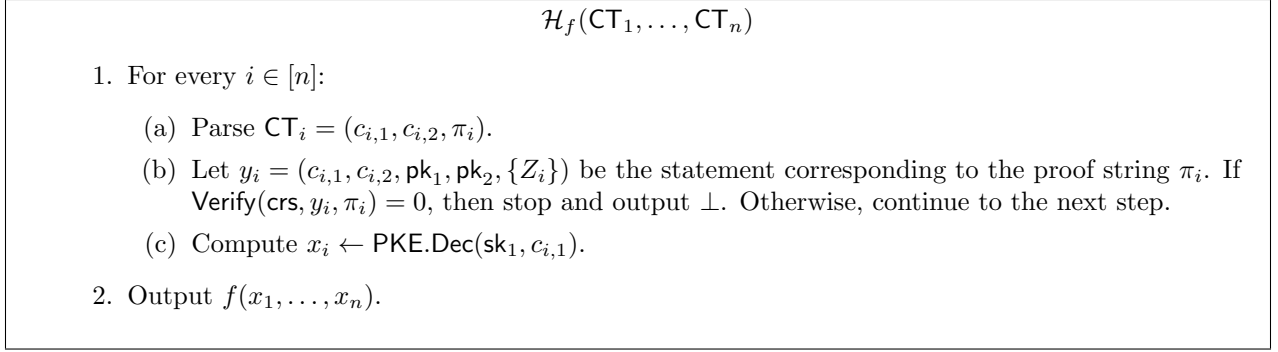


Figure 3: Functionality \mathcal{H}_f

The algorithm outputs SK_f as the secret key for f .

Size of Function \mathcal{H}_f . Similar to the construction in Section 4, in order to prove that \mathcal{FE}_{II} is $(0, q)$ -SIM-secure, we require the function \mathcal{H}_f to be padded with zeros such that the size of \mathcal{H}_f is equal to the size of its “simulated version” which has (among other things) $\binom{qn}{n}$ output values hardwired in it. Thus, in this case, the size of SK_f grows linearly with $\binom{qn}{n}$.

Note, however, that such a padding is *not* necessary to prove (t, q) -sel-IND-security for \mathcal{FE}_{II} . Indeed, in this case, the secret keys SK_f are independent of the number of message queries q made by the adversary.

Decryption $\text{FE.Dec}(\text{SK}_f, \text{CT}_1, \dots, \text{CT}_n)$: The decryption algorithm on input $(\text{CT}_1, \dots, \text{CT}_n)$ computes and outputs $\text{SK}_f(\text{CT}_1, \dots, \text{CT}_n)$.

This completes the description of \mathcal{FE}_{II} . The correctness property of the scheme follows from inspection.

Theorem 16. *Assuming differing-inputs obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme \mathcal{FE}_{II} is $(t, \text{poly}(k))$ -IND-secure for any $t \leq n$.*

We prove the above theorem in Appendix C. Further, we note that for $t = 0$ and $q = q(k)$ such that $\binom{qn}{n} = \text{poly}(k)$, our IND-security proof can be naturally extended to argue $(0, q)$ -SIM-security for \mathcal{FE}_{II} by using a similar simulation strategy as for our first construction (see Section 4). We formally state the claim below, but omit the proof details from this manuscript.

Theorem 17. *Let $q = q(k)$ be such that $\binom{qn}{n} = \text{poly}(k)$. Then, assuming differing-inputs obfuscation for all polynomial-time computable circuits and one-way functions, the proposed scheme \mathcal{FE}_{II} is $(0, q)$ -SIM-secure.*

6 Multi-Input Functional Encryption Implies Obfuscation

In this section, we prove that various flavors of multi-input FE imply well established notions of program obfuscation.

Indistinguishability Obfuscation from MI-FE. Our first result shows that the indistinguishability notion of multi-input FE unconditionally implies indistinguishability obfuscation (note that such an implication is not known to hold for single input FE). This, in particular, means that the use of indistinguishability obfuscation is unavoidable for multi-input FE, and, any future improvements in the complexity assumptions on which multi-input FE is based will only come with a corresponding improvement in the indistinguishability obfuscation constructions. We state the theorem below for the “weakest” case of *secret-key* multi-input functional encryption (this only strengthens our result).

Theorem 18. *(0, 2)-IND-secure MI-FE for general $(k+1)$ -ary functions unconditionally implies indistinguishability obfuscation for all circuits with k -bit inputs.*

Proof. We describe how to construct indistinguishability obfuscation for a circuit class \mathcal{C} where for every $C \in \mathcal{C}$, $C : \{0, 1\}^k \rightarrow \{0, 1\}^{k'}$ and $|C| = \ell$. Let \mathcal{FE} be a (0, 2)-IND-secure MI-FE scheme for general $(k+1)$ -ary functions. The PPT obfuscator $i\mathcal{O}$ works as follows.

- Consider a function g s.t. $g(x_1, \dots, x_k, C) = C(x_1 || \dots || x_k)$ where for all i , $x_i \in \{0, 1\}$, and, $C \in \{0, 1\}^\ell$. Observe that the function g acts as a universal circuit and treats its $(k+1)$ -th input as a circuit.
- The obfuscator $i\mathcal{O}$ first runs the setup algorithm for \mathcal{FE} to compute a master secret key MSK and encryption keys as $\text{EK}_1, \dots, \text{EK}_{k+1}$. It then runs the key generation algorithm of \mathcal{FE} to generate a secret key for the function g using MSK . Denote the resulting decryption key as SK_g .
- For all $i \in [k]$, $b \in \{0, 1\}$, let $\text{CT}_i^b \leftarrow \text{FE.Enc}(\text{EK}_i, b)$. All the encryptions are performed using independent random coins. Furthermore, let $\text{CT}_{k+1} \leftarrow \text{FE.Enc}(\text{EK}_{k+1}, C)$.
- The obfuscated circuit $i\mathcal{O}(C) = (\{\text{CT}_i^b\}_{b,i}, \text{CT}_{k+1}, \text{SK}_g)$.

To evaluate the obfuscated circuit on an input $x = (x_1, \dots, x_k)$, simply evaluate the decryption algorithm $\text{FE.Dec}(\text{SK}_g, \{\text{CT}_i^{x_i}\}_i, \text{CT}_{k+1})$. This results in $g(x_1, \dots, x_k, C) = C(x)$. This completes the description of the obfuscation scheme.

We now show that the above construction is indeed a secure indistinguishability obfuscation. This follows from the (0, 2)-IND-security of the underlying multi-input FE scheme. Consider any two functionally equivalent circuit C_0 and C_1 from \mathcal{C} . That is, for all $x \in \{0, 1\}^k$, $C_0(x) = C_1(x)$. Now, suppose for contradiction that there exists a PPT adversary \mathcal{A} that distinguishes between $i\mathcal{O}(C_0)$ and $i\mathcal{O}(C_1)$ with non-negligible advantage. We will construct an adversary \mathcal{B} that breaks (0, 2)-IND-security of \mathcal{FE} . The adversary \mathcal{B} runs \mathcal{A} and receives circuits C_0 and C_1 . It works as follows:

1. It defines challenge message vectors \vec{X}^0 and \vec{X}^1 , where $\vec{X}^b = \{x_{1,j}^b, \dots, x_{k+1,j}^b\}_{j \in [2]}$. For every $i \in [k]$, set $x_{i,1}^b = 0$ and $x_{i,2}^b = 1$. (Note here that $x_{i,j}^0 = x_{i,j}^1$.) Further, set $x_{k+1,1}^b = x_{k+1,2}^b = C_b$. \mathcal{B} sends over \vec{X}^0, \vec{X}^1 to the challenger in the IND security game. Let $\{\text{CT}_{1,j}, \dots, \text{CT}_{k+1,j}\}_{j \in [2]}$ denote the challenge ciphertexts received by \mathcal{B} .
2. Next, \mathcal{B} requests a secret key for the function g . Let SK_g be the secret key received by \mathcal{B} .
3. Now, \mathcal{B} sends over $(\{\text{CT}_{1,j}, \dots, \text{CT}_{k,j}\}_{j \in [2]}, \text{CT}_{k+1,1})$ as the challenge obfuscation to \mathcal{A} . \mathcal{B} simply outputs the guess b' returned by \mathcal{A} .

This completes the description of \mathcal{B} . We first argue that the challenge message vectors and the secret key query g are I-compatible as per IND security definition 3. (Here $I = \emptyset$.) To see this, note that for any $x = (x_1, \dots, x_k)$, we have that:

$$g(x_1, \dots, x_k, C_0) = g(x_1, \dots, x_k, C_1).$$

since C_0 and C_1 are functionally equivalent. Now, note that if the challenge bit b chosen by IND-security game challenge is equal to 0, then the resulting obfuscation is sent by \mathcal{B} to \mathcal{A} is of circuit C_0 ; otherwise it is an obfuscation of C_1 . Thus, if \mathcal{A} can distinguish between these two cases with non-negligible advantage, then \mathcal{B} also wins the IND game with the same advantage. This completes the proof. \square

Remark 19. *We remark that in the above proof, the “order” of the key query g is irrelevant. That is, g could be queried before or after the ciphertext queries.*

Virtual Black-Box Obfuscation from MI-FE. We now give two results for constructing virtual black-box obfuscation from various flavors of MI-FE. We first note that the same construction as above (Theorem 18), in fact, implies virtual black-box obfuscation, when \mathcal{FE} is $(0, 2)$ -SIM-secure.

Theorem 20. *$(0, 2)$ -SIM-secure FE for general $(k + 1)$ -ary functions unconditionally implies virtual black-box obfuscation for all circuits with k -bit inputs.*

Next, we show that SIM-secure multi-input FE, where at least one of the encryption keys may be made public, implies virtual black-box obfuscation.

Theorem 21. *$(1, 1)$ -SIM-secure MI-FE for general 2-ary functions unconditionally implies virtual black-box obfuscation for all circuits.*

Sketch. The proof of this theorem is quite similar to that of the previous one and we only provide a sketch here. The basic idea, as before, is to give out keys for a universal circuit g . The first input to g will be the function f which we wish to obfuscate. The encryption key EK_1 will be kept a secret, and, the ciphertext $\text{FE.Enc}(\text{EK}_1, f)$ will be included as part of the obfuscated circuit. The second input x will be the input on which the user wishes to evaluate f . Hence, the user is given access to the second encryption key EK_2 (as part of the obfuscated circuit) to enable it to encrypt any x . More details follow:

- Consider a function g s.t. $g(C, x) = C(x)$. Let \mathcal{FE} be a $(1, 1)$ -SIM-secure MI-FE for general 2-ary functions. The obfuscator VBB runs the setup algorithm for \mathcal{FE} to compute MSK and encryption keys $(\text{EK}_1, \text{EK}_2)$. It then runs the key generation algorithm of \mathcal{FE} to generate a secret key SK_g for the above function g using MSK.
- Let $\text{CT} \leftarrow \text{FE.Enc}(\text{EK}_1, C)$. The obfuscated circuit $\text{VBB}(C) = (\text{CT}, \text{EK}_2, \text{SK}_g)$.

To evaluate the obfuscated circuit on an input x , compute $\text{CT}' \leftarrow \text{FE.Enc}(\text{EK}_2, x)$, and, run $\text{FE.Dec}(\text{SK}_g, \text{CT}, \text{CT}')$. This results in $g(C, x) = C(x)$.

The virtual black-box obfuscation property follows from the fact that the view of the user can be simulated given access to a trusted party holding the first input f , and, evaluating $g(f, \cdot)$ on any second input x of user's choice. \square

6.1 Impossibility Results for SIM secure MI-FE

Here, we discuss some impossibility results for simulation secure MI-FE that complement our positive results given in Sections 4 and 5.

Recall that [BSW11, BO13] already establish the impossibility of $(0, \text{poly}(k))$ -SIM-secure functional encryption for 1-ary functions. We show that for n -ary functions, where $n \geq 2$, the

situation is much worse. In particular, recall that Barak et al. [BGI⁺01] proved an (unconditional) impossibility result for VBB obfuscation for general circuits. Then, combining their result with Theorem 21, we get the following result:

Theorem 22. *(1, 1)-SIM-secure multi-input functional encryption for general 2-ary functions is impossible.*

We remark that our positive results for SIM-secure MI-FE in Sections 4 and 5 are consistent with the above negative result and that of [BSW11, BO13].

Simulation secure MI-FE against Non-Adaptive Key Queries. So far in this paper, we have only considered simulation security for MI-FE in the setting where an adversary makes key queries *after* choosing the challenge messages. Following the terminology from the literature on single-input functional encryption, such queries are referred to as *adaptive* key queries. One can consider the “opposite” scenario, where the adversary is allowed to make key queries *before* choosing the challenge messages. This setting has been well studied in the case of single-input functional encryption, where such queries are referred to as *non-adaptive* key queries.

We now discuss the feasibility of simulation-based security for non-adaptive key queries (referred to as **NA-SIM** security) in our setting of multi-input FE. **NA-SIM** security for multi-input FE is defined similarly to definition 4, except that now the adversary is required to make key queries *before* (as opposed to *after*) choosing the challenge messages. More concretely, we can define (t, p, q) -**NA-SIM**-secure functional encryption where (as earlier) t denotes the number of encryption keys known to the adversary and q denotes the number of challenge messages per encryption key. The new parameter p denotes the total number of non-adaptive key queries by the adversary. For completeness, we provide a formal definition in Appendix D.

Now, observe that the proofs of Theorem 21 and Theorem 20 are insensitive to the “order” of the key query; i.e., they go through even if the key query is *non-adaptive*. Then, combining these results with the impossibility result of Barak et al [BGI⁺01], we obtain the following two (incomparable) results:

Theorem 23. *(1, 1, 1)-NA-SIM-secure multi-input functional encryption for general 2-ary functions is impossible.*

Theorem 24. *(0, 1, 2)-NA-SIM-secure multi-input functional encryption for general $(k + 1)$ -ary functions is impossible.*

We remark that we have stated Theorem 24 for the secret-key setting (as opposed to for general t) since it is the “weakest” case, and therefore only strengthens our result.

While the above impossibility results rule out achieving **NA-SIM**-security for general functions – in particular, they rule out **NA-SIM**-security for the arguably unnatural function that cannot be VBB obfuscated [BGI⁺01]) – we also provide another impossibility result for the weak pseudo-random function.

Let $\{F\}$ be a weak pseudo-random function family with key space K and message space X . The 2-ary $\text{wPRF}(\cdot, \cdot)$ functionality on input key $k \in K$ and message $x \in X$ outputs $F_K(x)$. We shall call k as the *first* input and x to be the *second* input to wPRF . We claim the following:

Theorem 25. *$(0, 1, \text{poly}(k))$ -NA-SIM-secure functional encryption for the weak PRF functionality $\text{wPRF}(\cdot, \cdot)$ is impossible.*

Proof. (Sketch). Here, we sketch a proof for black-box simulation. The proof follows along the same lines as in [AGVW13]. Suppose for contradiction that there exists a $(0, 1, \text{poly}(k))$ -NA-SIM-secure functional encryption \mathcal{FE} for the weak PRF functionality. Let $\ell - 1$ denote an upper bound on the ciphertext size in \mathcal{FE} . We construct an adversary \mathcal{A} that makes a single

key query and ℓ^2 number of message queries (per encryption key) such that every (black-box) simulator “fails” to simulate the view of \mathcal{A} .

The adversary \mathcal{A} first makes a single (non-adaptive) key query for the 2-ary function wPRF . Let $L = \ell^2$. Then, \mathcal{A} asks ciphertexts for L first inputs k_1, \dots, k_L and L second inputs x_1, \dots, x_L , where each k_i is chosen uniformly at random from the key space K and each x_i is chosen uniformly at random from the message space X . Now the simulator first needs to produce a key SK_{wPRF} and then it is given the functionality’s outputs $\{\text{wPRF}(k_i, x_j)\}_{i=1, j=1}^{L, L}$. Now, the simulator has to produce $2L$ ciphertexts $\{\text{CT}_i^1\}_{i=1}^L, \{\text{CT}_j^2\}_{j=1}^L$ such that for every $i \in [L], j \in [L]$, $\text{wPRF}(k_i, x_j) = \text{FE.Dec}(\text{SK}_f, \text{CT}_i^1, \text{CT}_j^2)$.

Thus, on the one hand, the simulator needs to “encode” all of the functionality’s outputs into $2L$ ciphertexts. On the other hand, the functionality’s outputs are $L^2 = \ell^4$ pseudo-random bits, while the total length of the $2L$ ciphertexts is $2L(\ell - 1) < 2\ell^3$ bits. Since a pseudo-random string cannot be efficiently compressed, we get a contradiction. \square

Discussion. Recall that the lower bounds of [AGVW13, CIJ⁺13] already establish that it is impossible to achieve $(0, \text{poly}(k), 1)$ -NA-SIM-secure functional encryption for 1-ary functions (specifically, the weak PRF functionality). That is, it is impossible to achieve NA-SIM security against an unbounded number of non-adaptive key queries even in the secret-key setting. Our impossibility results in Theorem 24 and Theorem 25 establish that it is also impossible to achieve NA-SIM security against an unbounded number of ciphertext queries. Thus, NA-SIM secure MFE is only possible for a *bounded* number of key queries and a *bounded* number of ciphertext queries. This is strictly worse than what can be achieved in the case of SIM security (where *unbounded* number of key queries can be achieved, in the secret-key setting, as exemplified by our positive results).

7 Extension to Randomized Functionalities

Our positive results for multi-input functional encryption presented in Sections 4 and 5 only concern with deterministic n -ary functions. Here, we discuss how to extend our results to handle *randomized* functionalities.

Modeling Security. In the single-input setting, the case of randomized functionalities was recently considered by Goyal et al. [GJKS13]. Very briefly, Goyal et al. observed that in the setting of randomized functionalities, the central challenge is to ensure that the random coins used for computing a function output are unbiased and remain hidden from the participants (i.e., the encryptor/sender and the decryptor/receiver). As such, in addition to requiring security against dishonest receivers, one must explicitly require security against dishonest senders to ensure that it is not possible to force “bad” outputs on an honest receiver.

We follow the same approach in our multi-input setting. Specifically, following [GJKS13], below we formalize a definition for security against dishonest senders. Overall, we will say that a multi-input functional encryption scheme for a randomized function family is secure if it achieves security against both dishonest senders and dishonest receivers.

Definition 26 (Security against Dishonest Senders). *We say that a functional encryption scheme \mathcal{FE} for n -ary (randomized) functions \mathcal{F} is t -secure against dishonest senders if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ such that the outputs of the following two experiments are computationally indistinguishable:*

Experiment $\text{REAL}_{\mathcal{A}}^{\mathcal{FE}}(1^k)$:

$(I, \text{st}_0) \leftarrow \mathcal{A}_0(1^k)$ where $|I| = t$
 $(\vec{E}, \text{MSK}) \leftarrow \text{FE.Setup}(1^k)$
 $(\{f\}, \mathcal{M}, \text{st}_1) \leftarrow \mathcal{A}_1(\text{st}_0, \vec{E})$
 $\text{SK}_f \leftarrow \text{FE.Keygen}(\text{MSK}, f) \forall f \in \{f\}$
 $\text{CT}_i \leftarrow \text{FE.Enc}(\text{EK}_i, x_i)$ where $x_i \leftarrow \mathcal{M} \forall i \in N \setminus I$
 $\alpha \leftarrow \mathcal{A}_2^{\mathcal{O}(\{\text{SK}_f\}, \vec{\text{CT}}, \cdot)}(\vec{\text{CT}}, \text{st}_1)$
Output: $(I, \{f\}, \mathcal{M}, \vec{x}, \{\text{out}\}, \alpha)$

Experiment $\text{IDEAL}_{\mathcal{S}}^{\mathcal{FE}}(1^k)$:

$(I, \text{st}_0) \leftarrow \mathcal{S}_0(1^k)$
 $(\{f\}, \mathcal{M}, \text{st}_1) \leftarrow \mathcal{S}_1(\text{st}_0)$
 $x_i \leftarrow \mathcal{M} \forall i \in N \setminus I$
 $\alpha \leftarrow \mathcal{S}_2^{\text{TP}(\{f\}, \vec{x}, \cdot)}(\text{st}_1)$
Output: $(I, \{f\}, \mathcal{M}, \vec{x}, \{\text{out}'\}, \alpha)$

where,

- In the real world, oracle $\mathcal{O}(\{\text{SK}_f\}, \vec{\text{CT}}, \cdot)$ accepts queries of the form $(\text{CT}_1^*, \dots, \text{CT}_t^*)$ such that for every $i \in [t]$, $j \in N \setminus I$, $\text{CT}_i^* \neq \text{CT}_j$. It outputs $\text{FE.Dec}(\text{SK}_f, \langle \text{CT}_1, \dots, \text{CT}_{n-t}, \text{CT}_1^*, \dots, \text{CT}_t^* \rangle)$ for every $\text{SK}_f \in \{\text{SK}_f\}$. Here, $\langle z_{i_1}, \dots, z_{i_n} \rangle$ denotes a permutation of the ciphertexts z_{i_1}, \dots, z_{i_n} such that z_{i_j} is mapped to the ℓ 'th location if z_{i_j} is the encrypted via the encryption key EK_ℓ . Further, $\{\text{out}\}$ denotes the set of outputs of \mathcal{O} to \mathcal{A}_2 's decryption queries.
- In the ideal world, the trusted party $\text{TP}(\{f\}, \vec{x}, \cdot)$ accepts queries of the form (x_1^*, \dots, x_t^*) and outputs $f_j(\langle x_{i_1}, \dots, x_{i_{n-t}}, x_1^*, \dots, x_t^* \rangle; r_j)$ for every $f_j \in \{f\}$. Here r_j is chosen uniformly at random and $\langle z_{i_1}, \dots, z_{i_n} \rangle$ denotes a permutation of the values z_{i_1}, \dots, z_{i_n} such that the value z_{i_j} is mapped to the ℓ 'th location if z_{i_j} is the ℓ 'th input (out of n inputs) to f . Further, $\{\text{out}'\}$ denotes the set of outputs of TP to the queries of \mathcal{S}_2 .

We now define SIM security for multi-input functional encryption for randomized functions. We note that IND security can be defined analogously; we skip the details.

Definition 27. We say that a functional encryption scheme \mathcal{FE} for n -ary (randomized) functions \mathcal{F} is (t_1, t_2, q) -SIM-secure if:

1. \mathcal{FE} is t_1 -secure against dishonest senders.
2. \mathcal{FE} is (t_2, q) -SIM-secure against dishonest receivers.

Positive Results for Randomized Functionalities. Building on the techniques of [GJKS13], both of our constructions for multi-input functional encryption presented in Sections 4 and 5 can be extended to handle randomized functionalities. Below, we outline the necessary modifications to our second scheme \mathcal{FE}_{II} to define a new scheme \mathcal{FE} . (We note that \mathcal{FE}_I can be modified in a similar manner to handle randomized functionalities.)

\mathcal{FE} is defined similarly to \mathcal{FE}_{II} , with the following necessary changes:

1. To encrypt a message x , we follow the same steps as in \mathcal{FE}_{II} to compute (c_1, c_2, π) . Next, we sample a key pair (sk, vk) for a strongly unforgeable one-time signature scheme. The final ciphertext CT consists of $(c_1, c_2, \pi, vk, \sigma)$, where σ is a signature over $c_1 \| c_2 \| \pi$ using sk .
2. To compute a secret key SK_f for a (randomized) function f , we first sample a key K for a puncturable pseudo-random function (PRF) [SW13, BW13, BGI13, KPTZ13]. Then, key SK_f is computed as $\text{diO}(\mathcal{H}'_f)$ where \mathcal{H}'_f is defined similarly to the functionality \mathcal{H}_f except that:
 - We additionally check whether the signature σ_i in each input ciphertext CT_i is valid.
 - Further, after decrypting each input ciphertext CT_i to compute x_i , we first compute randomness r as the output of the PRF on input $\text{CT}_1 \| \dots \| \text{CT}_n$ using key K . The final output is then computed as $f(x_1, \dots, x_n; r)$.

Very briefly, security against dishonest senders follows from the same ideas as in [NY90, DDN91, Sah99]. Specifically, incorporating the one-time signatures in the ciphertexts ensures that each ciphertext is unique (and therefore, an adversary cannot modify an honest sender's ciphertext to create a decryption query). Further, it is possible to extract the input from an adversarially created ciphertext using one of the secret keys (while using the semantic security for the other key). Security against dishonest receivers follows largely in the same manner as for \mathcal{FE}_{Π} . The main difference now is that (as in [GJKS13]), we use the punctured PRF to remove all the secret information in the PRF key for the point $\hat{C}T_1 \parallel \dots \parallel \hat{C}T_n$, where $\hat{C}T_1, \dots, \hat{C}T_n$ denotes a challenge ciphertext tuple. From the security of the obfuscation, it follows that this randomness remains hidden from a honest receiver. We refer the reader to [GJKS13] for more details on the proof.

References

- [ABG⁺13] Prabhanjan Ananth, Dan Boneh, Sanjam Garg, Amit Sahai, and Mark Zhandry. Differing-inputs obfuscation and applications. *IACR Cryptology ePrint Archive*, 2013:689, 2013.
- [AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO (2)*, 2013.
- [BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.
- [BCLO09] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and Adam O'Neill. Order-preserving symmetric encryption. In *EUROCRYPT*, 2009.
- [BCO11] Alexandra Boldyreva, Nathan Chenette, and Adam O'Neill. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In *CRYPTO*, 2011.
- [BCP13] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. *IACR Cryptology ePrint Archive*, 2013:650, 2013.
- [BGI⁺01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, 2001.
- [BGI13] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. *IACR Cryptology ePrint Archive*, 2013:401, 2013.
- [BO13] Mihir Bellare and Adam O'Neill. Semantically-secure functional encryption: Possibility results, impossibility results and the quest for a general definition. In *CANS*, 2013.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, 2011.
- [BW06] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.
- [BW07] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, 2007.
- [BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT*, 2013.

- [CIJ⁺13] Angelo De Caro, Vincenzo Iovino, Abhishek Jain, Adam O'Neill, Omer Paneth, and Giuseppe Persiano. On the achievability of simulation-based security for functional encryption. In *CRYPTO (2)*, 2013.
- [CKKC13] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Carlos Cid. Multi-client non-interactive verifiable computation. In *TCC*, 2013.
- [DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In *STOC*, pages 542–552, 1991.
- [DMNS06] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.
- [DNR⁺09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390, 2009.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [GGH⁺13a] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS*, 2013.
- [GGH⁺13b] Sanjam Garg, Craig Gentry, Shai Halevi, Amit Sahai, and Brent Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, 2013.
- [GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In *STOC*, pages 467–476, 2013.
- [GJKS13] Vipul Goyal, Abhishek Jain, Venkata Koppula, and Amit Sahai. How to compute randomized functions on encrypted data. *IACR Cryptology ePrint Archive*, 2013, 2013.
- [GKP⁺13a] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. How to run turing machines on encrypted data. In *CRYPTO (2)*, pages 536–553, 2013.
- [GKP⁺13b] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, 2013.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, 2006.
- [GS02] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.
- [GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, 2012.
- [GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In *STOC*, 2013.
- [KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. *IACR Cryptology ePrint Archive*, 2013:379, 2013.
- [KSW08] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, 2008.
- [LOS⁺10] Allison B. Lewko, Tatsuyuki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, 2010.

- [NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *STOC*, pages 427–437, 1990.
- [O’N10] Adam O’Neill. Definitional issues in functional encryption. *IACR Cryptology ePrint Archive*, 2010, 2010.
- [PR12] Omkant Pandey and Yannis Rouselakis. Property preserving symmetric encryption. In *EUROCRYPT*, 2012.
- [PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC*, 2012.
- [Sah99] Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *FOCS*, pages 543–553, 1999.
- [SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *ACM Conference on Computer and Communications Security*, pages 463–472, 2010.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, 2005.
- [SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *IACR Cryptology ePrint Archive*, 2013:454, 2013.

A Completing sel-IND Security Proof for \mathcal{FE}_1

Lemma 28 ($H_0 \stackrel{c}{\equiv} H_1$). *Assuming that Com is a (computationally) hiding commitment scheme, the outputs of experiments H_0 and H_1 are computationally indistinguishable.*

Proof. Recall that the only difference between H_0 and H_1 is the manner in which the commitments $\{Z_1^{i,j}\}$ are computed: in H_0 , every $Z_1^{i,j}$ is a commitment to the all zeros string 0^{len} , while in H_1 , $Z_1^{i,j}$ is a commitment to $\hat{c}_1^{i,j} \parallel \hat{c}_2^{i,j}$. Further, note that the randomness used to compute $Z_1^{i,j}$ is not used elsewhere in the experiment. Then, by a standard hybrid argument, the indistinguishability of H_0 and H_1 follows from the computational hiding property of Com. We omit the details. \square

Lemma 29 ($H_1 \stackrel{c}{\equiv} H_2$). *Assuming that (CRSGen, Prove, Verify) is witness indistinguishable, the outputs of experiments H_1 and H_2 are computationally indistinguishable.*

Proof. Recall that the only difference between H_1 and H_2 is the manner in which the proof strings $\hat{\pi}^{i,j}$ in challenge ciphertexts $\hat{CT}_{i,j}$ are computed: in H_1 , every $\hat{\pi}^{i,j}$ is computed using the *real* witness, while in H_2 , $\hat{\pi}^{i,j}$ is computed using the *trapdoor* witness. Then, by a standard hybrid argument, the indistinguishability of H_1 and H_2 follows from the witness indistinguishability property of the NIWI proof system. \square

Lemma 30 ($H_2 \stackrel{c}{\equiv} H_3$). *Assuming that Com is a (computationally) hiding commitment scheme, the outputs of experiments H_2 and H_3 are computationally indistinguishable.*

Proof. Recall that the only difference between H_2 and H_3 is the manner in which the commitments $\{Z_2^i\}$ are computed: in H_2 , every Z_2^i , where $i \in I$, is a commitment to 0, while in H_3 , Z_2^i is a commitment to 1. Further, note that the randomness used to compute Z_2^i is not used anywhere else in the experiment. Then, by a standard hybrid argument, the indistinguishability of H_2 and H_3 follows from the computational hiding property of Com. \square

Lemma 31 ($H_3 \stackrel{c}{=} H_4$). *Assuming that $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ is a semantically-secure public-key encryption scheme, the outputs of experiments H_3 and H_4 are computationally indistinguishable.*

Proof. Recall that the only difference between H_3 and H_4 is the manner in which the second ciphertexts $\hat{c}_2^{i,j}$ in the challenge ciphertexts $\hat{\text{CT}}_{i,j}$ are computed: in H_3 , $\hat{c}_2^{i,j}$ is an encryption of the challenge message $x_{i,j}$, while in H_4 , $\hat{c}_2^{i,j}$ is an encryption of 0. Further, note that neither the randomness $s_2^{i,j}$ used to compute $\hat{c}_2^{i,j}$, nor the secret key sk_2 is used anywhere else in the experiment. Then, by a standard hybrid argument, the indistinguishability of H_3 and H_4 follows from the semantic security of PKE. \square

Lemma 32 ($H_4 \stackrel{c}{=} H_5$). *Assuming that $i\mathcal{O}$ is an indistinguishability obfuscator, Com is perfectly binding and $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a proof system, the outputs of the experiments H_4 and H_5 are computationally indistinguishable.*

Proof. We prove the lemma for the simplified case where the adversary makes a single key query f . We remark that by a standard hybrid argument, the proof can be easily extended to the more general case where the adversary makes $\text{poly}(k)$ number of key queries.

Now, note that the only difference between H_4 and H_5 is the manner in which the secret key SK_f for the key query f is computed: in experiment H_4 , SK_f is an indistinguishability obfuscation of G_f , while in H_5 , SK_f is an indistinguishability obfuscation of $\text{Sim}.G'_f$. Now, if G_f and G'_f have the same output behavior on *all* input points, then the computational indistinguishability of H_4 and H_5 follows immediately from the indistinguishability of $i\mathcal{O}(G_f)$ and $i\mathcal{O}(G'_f)$. Thus, all that remains to prove is that for all inputs z , $G_f(z) = G'_f(z)$.

Towards that end, we first assume without loss of generality that the encryption scheme $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ does not have any decryption error. We make the following claim:

Claim 33. *For any input z , $G_f(z) = \perp$ iff $G'_f(z) = \perp$.*

Proof. Let $z = (\text{CT}_1, \dots, \text{CT}_n)$ be any input to G_f and G'_f . For every $i \in [n]$, let $\text{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$. Note that both $\text{Sim}.G_f$ and $\text{Sim}.G'_f$ output \perp on input z iff there exists $i \in [n]$ such that $\text{Verify}(\text{crs}, y_i, \pi_i) = 0$, where $y_i = (c_{i,1}, c_{i,2}, \text{pk}_1, \text{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$ is the statement corresponding to the NIWI proof π_i . The claim follows. \square

Following the above claim, we shall call an input z to G_f and G'_f to be a *valid* input if $G_f(z) \neq \perp$ (and $G'_f(z) \neq \perp$). We now demonstrate that the outputs of G_f and G'_f differ on a valid input z only if z satisfies some specific properties. Later, we will rely on the binding property of Com and the statistical soundness of the NIWI proof system to show that such an input z does not exist, thus completing the proof.

Claim 34. *Let $\{\hat{\text{CT}}_{1,j}, \dots, \hat{\text{CT}}_{n,j}\}_{j=1}^q$ denote the challenge ciphertexts given to the adversary, where every $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Then, for every valid input $z = (\text{CT}_1, \dots, \text{CT}_n)$ to G_f and G'_f such that $G_f(z) \neq G'_f(z)$, there exists $i \in [n]$, $\text{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$ in z such that one of the following two cases holds:*

Case 1: *If $i \in I$, then $c_{i,1}$ and $c_{i,2}$ are encryptions of different messages, and for every $j \in [q]$, either $\hat{c}_{i,1} \neq \hat{c}_1^{i,j}$ or $c_{i,2} \neq \hat{c}_2^{i,j}$.*

Case 2: *If $i \in N \setminus I$, then for every $j \in [q]$, either $c_{i,1} \neq \hat{c}_1^{i,j}$ or $c_{i,2} \neq \hat{c}_2^{i,j}$.*

Proof. Suppose that the claim is false. That is, there exists a valid input $z^* = (\text{CT}_1^*, \dots, \text{CT}_n^*)$ such that $G_f(z^*) \neq G'_f(z^*)$, yet z satisfies the following conditions:

Condition A: For every $i \in I$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ is such that:

1. Either there exists $j_i \in [q]$ such that $c_{i,1}^* = \hat{c}_1^{i,j_i}$ and $c_{i,2}^* = \hat{c}_2^{i,j_i}$, or
2. $c_{i,1}^*$ and $c_{i,2}^*$ are encryptions of the *same* message. Let x'_i denote this message.

Condition B: For every $i \in N \setminus I$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$, there exists $j_i \in [q]$ such that $c_{i,1}^* = \hat{c}_1^{i,j_i}$ and $c_{i,2}^* = \hat{c}_2^{i,j_i}$.

Let us inspect the outputs of G_f and G'_f on the input z^* . We have that:

$$G_f(z^*) = f \left(\left\langle \left\{ \text{PKE.Dec}(\text{sk}_1, \hat{c}_1^{i,j_i}) \right\}_{i \in N \setminus I}, \left\{ \text{PKE.Dec}(\text{sk}_1, c_{i,1}^*) \right\}_{i \in I} \right\rangle \right),$$

$$G'_f(z^*) = f \left(\left\langle \left\{ \text{PKE.Dec}(\text{sk}_2, \hat{c}_2^{i,j_i}) \right\}_{i \in N \setminus I}, \left\{ \text{PKE.Dec}(\text{sk}_2, c_{i,2}^*) \right\}_{i \in I} \right\rangle \right),$$

where for $\ell \in [2]$, $\left\langle \left\{ \text{PKE.Dec}(\text{sk}_\ell, \hat{c}_\ell^{i,j_i}) \right\}_{i \in N \setminus I}, \left\{ \text{PKE.Dec}(\text{sk}_\ell, c_{i,\ell}^*) \right\}_{i \in I} \right\rangle$ denotes the “arrangement” of the values $\left\{ \text{PKE.Dec}(\text{sk}_\ell, \hat{c}_\ell^{i,j_i}) \right\}_{i \in N \setminus I}, \left\{ \text{PKE.Dec}(\text{sk}_\ell, c_{i,\ell}^*) \right\}_{i \in I}$ according to their input positions in f . Now, let $I' \subseteq I$ be such that for every $i \in I'$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ satisfies the condition A(2). Thus, for every $i \in I \setminus I'$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ satisfies the condition A(1). Then, we have:

$$G_f(z^*) = f \left(\left\langle \left\{ x_{i,j_i}^0 \right\}_{i \in N \setminus I'}, \left\{ x'_i \right\}_{i \in I'} \right\rangle \right); G'_f(z^*) = f \left(\left\langle \left\{ x_{i,j_i}^1 \right\}_{i \in N \setminus I'}, \left\{ x'_i \right\}_{i \in I'} \right\rangle \right),$$

where $\vec{X}^0 = \{x_{1,j}^0, \dots, x_{n,j}^0\}_{j=1}^q$ and $\vec{X}^1 = \{x_{1,j}^1, \dots, x_{n,j}^1\}_{j=1}^q$ are the challenge messages.

Now, it follows from the I-Compatibility property in the IND-security definition (see Definition 3) that $G_f(z^*) = G'_f(z^*)$, which is a contradiction. \square

Completing the proof of Lemma 32. We now prove that for every valid input z , $G_f(z) \neq G'_f(z)$. For the sake of contradiction, suppose not. That is, let z^* be a valid input such that $G_f(z^*) = G'_f(z^*)$. Following Claim 34, fix $i \in [n]$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ in z^* to be such that either Case 1 or Case 2 holds.

First observe that since z^* is a valid input, we have that $\text{Verify}(\text{crs}, y_i^*, \pi_i^*) = 1$, where $y_i^* = (c_{i,1}^*, c_{i,2}^*, \text{pk}_1, \text{pk}_2, \{Z_1^{i,j}\}, Z_1^i)$ is the statement corresponding to the proof string π_i^* . Then, since $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a statistically sound proof system, it follows that the statement y_i^* must be *true*, i.e., either there exists a *real* witness or a *trapdoor* witness for y_i^* (see Section 4 for the definitions of real and trapdoor witnesses). We now consider the two cases:

Case 1. $i \in I$: Since $c_{i,1}^*$ and $c_{i,2}^*$ are encryptions of different messages, there does not exist a *real* witness for y_i^* . Then, suppose that there exists a *trapdoor* witness $w_{\text{trap}} = (j, r_1^{i,j})$ for y_i^* . That is, suppose that $\exists j \in [q]$ and randomness $r_1^{i,j}$ such that $Z_1^{i,j} = \text{Com}(c_{i,1}^*, c_{i,2}^*; r_1^{i,j})$. However, note that in experiments H_4 and H_5 , $Z_1^{i,j}$ is computed as a commitment to $\hat{c}_1^{i,j} \parallel \hat{c}_2^{i,j}$. Since Com is perfectly binding and either $c_{i,1}^* \neq \hat{c}_1^{i,j}$ or $c_{i,2}^* \neq \hat{c}_2^{i,j}$, we obtain a contradiction.

Case 2. $i \in N \setminus I$: First observe that since Z_2^i is computed as a commitment to 1 in experiments H_4 and H_5 , it follows from the perfect binding property of Com that there does not exist a *real* witness for y_i^* . Then, suppose that there exists a *trapdoor* witness $w_{\text{trap}} = (j, r_1^{i,j})$ for y_i^* . That is, suppose that $\exists j \in [q]$ and randomness $r_1^{i,j}$ such that $Z_1^{i,j} = \text{Com}(c_{i,1}^*, c_{i,2}^*; r_1^{i,j})$. However, note that in experiments H_4 and H_5 , $Z_1^{i,j}$ is computed as a commitment to $\hat{c}_1^{i,j} \parallel \hat{c}_2^{i,j}$. Since Com is perfectly binding and either $c_{i,1}^* \neq \hat{c}_1^{i,j}$ or $c_{i,2}^* \neq \hat{c}_2^{i,j}$, we obtain a contradiction. \square

Lemma 35 ($H_5 \stackrel{c}{\equiv} H_6$). *Assuming that $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ is a semantically-secure public-key encryption scheme, the outputs of experiments H_5 and H_6 are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 31. \square

Lemma 36 ($H_6 \stackrel{c}{\equiv} H_7$). *Assuming that $i\mathcal{O}$ is an indistinguishability obfuscator, $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a proof system, and Com is perfectly binding, the outputs of experiments H_6 and H_7 are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 32. \square

Lemma 37 ($H_7 \stackrel{c}{\equiv} H_8$). *Assuming that Com is a (computationally) hiding commitment scheme, the outputs of experiments H_7 and H_8 are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 30. \square

Lemma 38 ($H_8 \stackrel{c}{\equiv} H_9$). *Assuming that $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is witness indistinguishable, the outputs of experiments H_8 and H_9 are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 29. \square

Lemma 39 ($H_9 \stackrel{c}{\equiv} H_{10}$). *Assuming that Com is a (computationally) hiding commitment scheme, the outputs of experiments H_9 and H_{10} are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 28. \square

B Completing SIM Security Proof for \mathcal{FE}_1

We now describe a series of hybrid experiments H_0, \dots, H_8 , where H_0 corresponds to the real world and H_8 corresponds to the ideal world experiment. For every i , we will prove that the output of H_i is computationally indistinguishable from the output of H_{i+1} .

Hybrid H_0 : This is the real experiment.

Hybrid H_1 : This experiment is the same as H_0 except in the manner in which the key queries of the adversary are answered. Let $\{x_{1,j}, \dots, x_{n,j}\}_{j=1}^q \leftarrow M$ be the challenge messages. Then, whenever the adversary makes a key query f , we perform the following steps:

- Query the trusted party TP on function f . For every $j_1, \dots, j_n \in [q]$, the trusted party computes and returns the function output $\text{out}[j_1, \dots, j_n] = f(x_{1,j_1}, \dots, x_{n,j_n})$.
- Compute the secret key SK_f for function f as $\text{SK}_f \leftarrow i\mathcal{O}(\text{Sim.G}_f)$, where Sim.G_f is as described in Figure 2.

For every $i \in [n]$, $j \in [q]$, let $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$ denote the challenge ciphertext computed by the experiment. Then, note that Sim.G_f has the master secret key MSK , the ciphertext pairs $\{\hat{c}_1^{i,j}, \hat{c}_2^{i,j}\}$ and the outputs $\{\text{out}[j_1, \dots, j_n]\}$ hardwired in it.

Hybrid H_2 : This experiment is the same as H_1 except that the setup algorithm computes the commitments $\{Z_1^{i,j}\}$ in the following manner: let the challenge ciphertext $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Then, $Z_1^{i,j} \leftarrow \text{Com}(\hat{c}_1^{i,j} \parallel \hat{c}_2^{i,j})$.

Hybrid H_3 : This experiment is the same as H_2 except that in every challenge ciphertext $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the proof string $\hat{\pi}^{i,j}$ is computed using the *trapdoor* witness.

Hybrid H_4 : This experiment is the same as H_3 except that the setup algorithm computes every Z_2^i as a commitment to 1 (instead of 0). That is, for every $i \in [n]$, $Z_2^i \leftarrow \text{Com}(1)$.

Hybrid H₅: This experiment is the same as H₄ except that in every challenge ciphertext $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *second* ciphertext $\hat{c}_2^{i,j}$ is an encryption of zeros, i.e., $\hat{c}_2^{i,j} \leftarrow \text{FE.Enc}(\text{EK}_i, 0^k)$.

Hybrid H₆: This experiment is the same as H₅ except that for every key query f , the corresponding secret key SK_f is computed as $\text{SK}_f \leftarrow i\mathcal{O}(\text{Sim.G}'_f)$ where $\text{Sim.G}'_f$ is the same as the function Sim.G_f except that:

1. It has secret key sk_2 hardwired instead of sk_1 .
2. It decrypts the *second* component of each input ciphertext using sk_2 . More concretely, in step 1(c), plaintext x'_i is computed as $x'_i \leftarrow \text{PKE.Dec}(\text{sk}_2, c_{i,2})$.

Hybrid H₇: This experiment is the same as H₆ except that in every challenge ciphertext $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *first* ciphertext $\hat{c}_1^{i,j}$ is an encryption of zeros, i.e., $\hat{c}_1^{i,j} \leftarrow \text{FE.Enc}(\text{EK}_i, 0^k)$.

Hybrid H₈: This experiment is the same as H₇ except that for every key query f , the corresponding secret key SK_f is computed as $\text{SK}_f \leftarrow i\mathcal{O}(\text{Sim.G}_f)$. Note that this is the ideal world experiment.

This completes the description of the hybrid experiments. We note that the proof of indistinguishability of the hybrid experiments described above bear much similarity to the proof of IND security (Section 4.1). Therefore, to avoid repetition, below we only focus on the key hybrids that differ from the IND security case. Specifically, below, we prove indistinguishability of hybrid experiments H₀ and H₁, and then H₅ and H₆. For details on the rest of the proof, see Appendix A.

Lemma 40 ($\text{H}_0 \stackrel{c}{\equiv} \text{H}_1$). *Assuming that $i\mathcal{O}$ is an indistinguishability obfuscator, the outputs of experiments H₀ and H₁ are computationally indistinguishable.*

Proof. We prove the lemma for the simplified case where the adversary makes a single key query f . By a standard hybrid argument, the proof can be easily extended to the more general case where the adversary makes $\text{poly}(k)$ number of key queries.

Now, note that the only difference between H₀ and H₁ is the manner in which the secret key SK_f for the key query f is computed: in experiment H₀, SK_f is an indistinguishability obfuscation of G_f , while in H₁, SK_f is an indistinguishability obfuscation of Sim.G_f . Now, if G_f and Sim.G_f have the same output behavior on *all* input points, then the computational indistinguishability of H₀ and H₁ follows immediately from the indistinguishability of $i\mathcal{O}(\text{G}_f)$ and $i\mathcal{O}(\text{Sim.G}_f)$. Thus, all that remains to prove is that for all inputs z , $\text{G}_f(z) = \text{Sim.G}_f(z)$.

Towards that end, let $\{\hat{\text{CT}}_{1,j}, \dots, \hat{\text{CT}}_{n,j}\}_{j=1}^q$ denote the challenge ciphertexts computed in experiments H₁ and H₂, where every $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. We say that an input $z = (\text{CT}_1, \dots, \text{CT}_n)$ to G_f and Sim.G_f is *special* if for every $\text{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$:

- The proof π_i is accepting, and
- There exists $j_i \in [q]$ s.t. $c_{i,1} = \hat{c}_1^{i,j_i}$ and $c_{i,2} = \hat{c}_2^{i,j_i}$.

Further, we call (j_1, \dots, j_n) to be the “index set” of z .

Now note that the only difference between the functions G_f and Sim.G_f is that on a special input z with index set (j_1, \dots, j_n) , Sim.G_f skips the usual decryption step and directly outputs the value $\text{out}[j_1, \dots, j_n]$ hardwired in its description. Recall that (by definition) $\text{out}[j_1, \dots, j_n] = f(x_{1,j_1}, \dots, x_{n,j_n})$ where $\{x_{1,j}, \dots, x_{n,j}\}_{j=1}^q$ denote the challenge messages. However, on such an input z , by performing the decryption step, G_f obtains the messages $(x_{1,j_1}, \dots, x_{n,j_n})$ and therefore its output is $f(x_{1,j_1}, \dots, x_{n,j_n})$ as well. \square

Lemma 41 ($H_5 \stackrel{c}{=} H_6$). *Assuming that $i\mathcal{O}$ is an indistinguishability obfuscator, Com is perfectly binding, and $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a proof system, the outputs of experiments H_5 and H_6 are computationally indistinguishable.*

Proof. We prove the lemma for the simplified case where the adversary makes a single key query f . By a standard hybrid argument, our proof can be easily extended to the more general case where the adversary makes $\text{poly}(k)$ number of key queries.

Now, note that the only difference between H_5 and H_6 is the manner in which the secret key SK_f for the key query f is computed: in experiment H_5 , SK_f is an indistinguishability obfuscation of Sim.G_f , while in H_6 , SK_f is an indistinguishability obfuscation of $\text{Sim.G}'_f$. Now, if Sim.G_f and $\text{Sim.G}'_f$ have the same output behavior on *all* input points, then the computational indistinguishability of H_5 and H_6 follows immediately from the indistinguishability of $i\mathcal{O}(\text{Sim.G}_f)$ and $i\mathcal{O}(\text{Sim.G}'_f)$. Thus, all that remains to prove is that for all inputs z , $\text{Sim.G}_f(z) = \text{Sim.G}'_f(z)$.

Towards that end, we first assume without loss of generality that the encryption scheme $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ does not have any decryption error. We make the following claim:

Claim 42. *For any input z , $\text{Sim.G}_f(z) = \perp$ iff $\text{Sim.G}'_f(z) = \perp$.*

Proof. Let $z = (\text{CT}_1, \dots, \text{CT}_n)$ be any input to Sim.G_f and $\text{Sim.G}'_f$. For every $i \in [n]$, let $\text{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$. Note that both Sim.G_f and $\text{Sim.G}'_f$ output \perp on input z iff there exists $i \in [n]$ such that $\text{Verify}(\text{crs}, y_i, \pi_i) = 0$, where $y_i = (c_{i,1}, c_{i,2}, \text{pk}_1, \text{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$ is the statement corresponding to the NIWI proof π_i . The claim immediately follows. \square

Following the above claim, we shall call an input z to Sim.G_f and $\text{Sim.G}'_f$ to be a *valid* input if $\text{Sim.G}_f(z) \neq \perp$ (and $\text{Sim.G}'_f(z) \neq \perp$). We make the following claim regarding valid inputs:

Claim 43. *Let $\{\hat{\text{CT}}_{1,j}, \dots, \hat{\text{CT}}_{n,j}\}_{j=1}^q$ denote the challenge ciphertexts given to the adversary, where every $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Let $z = (\text{CT}_1, \dots, \text{CT}_n)$ denote a valid input to Sim.G_f and $\text{Sim.G}'_f$. Then, for every $\text{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$, there exists $j_i \in [q]$ s.t. $c_{i,1} = \hat{c}_1^{i,j_i}$ and $c_{i,2} = \hat{c}_2^{i,j_i}$.*

Proof. Suppose that the claim is false. That is, for a valid input $z = (\text{CT}_1, \dots, \text{CT}_n)$, $\exists \text{CT}_i = (c_{i,1}, c_{i,2}, \pi_i)$ s.t. $\forall j \in [q]$, either $c_{i,1} \neq \hat{c}_1^{i,j}$ or $c_{i,2} \neq \hat{c}_2^{i,j}$. Now, since z is a valid input, we have that $\text{Verify}(\text{crs}, y_i, \pi_i) = 1$, where $y_i = (c_{i,1}, c_{i,2}, \text{pk}_1, \text{pk}_2, \{Z_1^{i,j}\}, Z_2^i)$ is the statement corresponding to the NIWI proof π_i . Then, since $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a statistically sound proof system, it follows that the statement y_i must be *true*. We consider the following two cases:

Case 1: The ciphertexts $c_{i,1}$ and $c_{i,2}$ are encryptions of the same message and there exists randomness r_2^i s.t. $Z_2^i \leftarrow \text{Com}(0; r_2^i)$. However, note that in experiments H_5 and H_6 , Z_2^i is computed as a commitment to 1. Since Com is a perfectly binding commitment scheme, we obtain a contradiction.

Case 2: $\exists j \in [q]$ and randomness $r_1^{i,j}$ such that $Z_1^{i,j} = \text{Com}(c_{i,1} \| c_{i,2}; r_1^{i,j})$. However, note that in experiments H_5 and H_6 , $Z_1^{i,j}$ is computed as a commitment to $\hat{c}_1^{i,j} \| \hat{c}_2^{i,j}$. Since Com is a perfectly binding commitment scheme and either $c_{i,1} \neq \hat{c}_1^{i,j}$ or $c_{i,2} \neq \hat{c}_2^{i,j}$, we obtain a contradiction.

This completes the proof of the above claim. \square

Completing the proof of Lemma 41. Following Claim 42, we only need to prove that for every valid input z , $\text{Sim.G}_f(z) = \text{Sim.G}'_f(z)$. Now, let $z = (\text{CT}_1, \dots, \text{CT}_n)$ be any valid input to Sim.G_f and $\text{Sim.G}'_f$. From Claim 49, we have that for every $i \in [n]$, there exists $j_i \in [q]$ such that $\text{CT}_i = (\hat{c}_1^{i,j_i}, \hat{c}_2^{i,j_i}, \pi_i)$. Then, note that on such an input $z = (\text{CT}_1, \dots, \text{CT}_n)$, both Sim.G_f and $\text{Sim.G}'_f$ output the *same* (programmed) value, i.e., $\text{out}[j_1, \dots, j_n]$.

This completes the proof of Lemma 41. \square

C Proving IND Security for \mathcal{FE}_{\parallel}

We now prove that the proposed scheme \mathcal{FE}_{\parallel} is $(t, \text{poly}(k))$ -IND-secure for any $t \leq n$ and arbitrary $\text{poly}(k)$ number of message queries. We will prove security via a series of hybrid experiments H_0, \dots, H_8 , where H_0 (resp., H_8) corresponds to the real world experiment with challenge bit $b = 0$ (resp., $b = 1$).

Hybrid H_0 : This is the real experiment with challenge bit $b = 0$.

Hybrid H_1 : This experiment is the same as H_0 except that the setup algorithm computes a “simulated” CRS for the simulation-sound NIZK proof system, i.e., the CRS is computed as $(\text{crs}, \tau) \leftarrow \text{Sim.CRSGen}(1^k)$.

Hybrid H_2 : This experiment is the same as H_1 except that in every challenge ciphertext $\hat{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, $\hat{\pi}^{i,j}$ is computed as a simulated proof, i.e., $\hat{\pi}^{i,j} \leftarrow \text{Sim.Prove}(\text{crs}, \tau, y_{i,j})$ where the statement $y_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \text{pk}_1, \text{pk}_2, Z_i)$.

Hybrid H_3 : This experiment is the same as H_2 except that for every $i \in \mathbb{N} \setminus \mathbb{I}$, the setup algorithm computes every Z_i as a commitment to 1 (instead of 0), i.e., $Z_i \leftarrow \text{Com}(1)$.

Hybrid H_4 : This experiment is the same as H_3 except that in every challenge ciphertext $\hat{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *second* ciphertext $\hat{c}_2^{i,j}$ is an encryption of the challenge message $x_{i,j}^1$ (as opposed to $x_{i,j}^0$), i.e., $\hat{c}_2^{i,j} \leftarrow \text{FE.Enc}(\text{EK}_i, x_{i,j}^1)$.

Hybrid H_5 : This experiment is the same as H_4 except that for every key query f , the corresponding secret key SK_f is computed as $\text{SK}_f \leftarrow \text{diO}(\mathcal{H}'_f)$ where \mathcal{H}'_f is the same as the function \mathcal{H}_f except that:

1. It has secret key sk_2 hardwired instead of sk_1 .
2. It decrypts the *second* component of each input ciphertext using sk_2 . More concretely, in step 1(c), plaintext x'_i is computed as $x'_i \leftarrow \text{PKE.Dec}(\text{sk}_2, c_{i,2})$.

Hybrid H_6 : This experiment is the same as H_5 except that in every challenge ciphertext $\hat{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the *first* ciphertext $\hat{c}_1^{i,j}$ is an encryption of the challenge message $x_{i,j}^1$ (as opposed to $x_{i,j}^0$), i.e., $\hat{c}_1^{i,j} \leftarrow \text{FE.Enc}(\text{EK}_i, x_{i,j}^1)$.

Hybrid H_7 : This experiment is the same as H_6 except that for every key query f , the corresponding secret key SK_f is computed as $\text{SK}_f \leftarrow \text{diO}(\mathcal{H}_f)$.

Hybrid H_8 : This experiment is the same as H_7 except that the setup algorithm computes every Z_i as a commitment to 0.

Hybrid H_9 : This experiment is the same as H_8 except that in every challenge ciphertext $\hat{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$, the proof string $\hat{\pi}^{i,j}$ is computed using the honest prover algorithm.

Hybrid H_{10} : This experiment is the same as H_9 except that the setup algorithm computes an “honest” CRS for the NIZK proof system, i.e., the CRS is computed as $\text{crs} \leftarrow \text{CRSGen}(1^k)$. Note that this is the real experiment with challenge bit $b = 1$.

This completes the description of the hybrids. We now prove their computational indistinguishability via a series of lemmas.

Lemma 44 ($H_0 \stackrel{c}{=} H_1$). *Assuming that $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a zero-knowledge argument system, the outputs of experiments H_0 and H_1 are computationally indistinguishable.*

Proof. This follows immediately from the fact that the distributions $\{\text{CRSGen}(1^k)\}$ and $\{\text{Sim.CRSGen}(1^k)\}$ are computationally indistinguishable. \square

Lemma 45 ($H_1 \stackrel{c}{=} H_2$). *Assuming that $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a zero-knowledge argument system, the outputs of experiments H_1 and H_2 are computationally indistinguishable.*

Proof. Recall that the only difference between H_1 and H_2 is the manner in which the proof strings $\hat{\pi}^{i,j}$ in challenge ciphertexts $\hat{\text{CT}}_{i,j}$ are computed: in H_1 , every $\hat{\pi}^{i,j}$ is computed honestly using the witness, while in H_2 , $\hat{\pi}^{i,j}$ is a simulated proof computed using the simulator for the NIZK argument system. Then, by a standard hybrid argument, the indistinguishability of H_2 and H_3 follows from the zero-knowledge property of the NIZK argument system. \square

Lemma 46 ($H_2 \stackrel{c}{=} H_3$). *Assuming that Com is a computationally hiding commitment scheme, the outputs of experiments H_2 and H_3 are computationally indistinguishable.*

Proof. Recall that the only difference between H_2 and H_3 is the manner in which the commitment $\{Z_i\}_{i \in \mathbb{N} \setminus I}$ are computed: in H_2 , every Z_i is a commitment to 0, while in H_3 , Z_i is a commitment to 1. Further, note that the randomness used to compute Z is not used anywhere else in the experiment. Then, the indistinguishability of H_2 and H_3 follows immediately from the computational hiding property of Com . \square

Lemma 47 ($H_3 \stackrel{c}{=} H_4$). *Assuming that $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ is a semantically-secure public-key encryption scheme, the outputs of experiments H_3 and H_4 are computationally indistinguishable.*

Proof. Recall that the only difference between H_3 and H_4 is the manner in which the second ciphertexts $\hat{c}_2^{i,j}$ in the challenge ciphertexts $\hat{\text{CT}}_{i,j}$ are computed: in H_3 , $\hat{c}_2^{i,j}$ is an encryption of the challenge message $x_{i,j}^0$, while in H_4 , $\hat{c}_2^{i,j}$ is an encryption of $x_{i,j}^1$. Further, note that neither the randomness $s_2^{i,j}$ used to compute $\hat{c}_2^{i,j}$ nor the secret key sk_2 is used anywhere else in the experiment. Then, by a standard hybrid argument, the indistinguishability of H_3 and H_4 follows from the semantic security of PKE. \square

Lemma 48 ($H_4 \stackrel{c}{=} H_5$). *Assuming that diO is a differing-inputs obfuscator, Com is perfectly binding and $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is simulation-sound, the outputs of experiments H_4 and H_5 are computationally indistinguishable.*

Proof. We prove the lemma for the simplified case where the adversary makes a single key query f . This query could either be made by adversary \mathcal{A}_1 or \mathcal{A}_2 . In the former case, we refer to f as a *non-adaptive* key query, while in the latter case, we refer to it as an *adaptive* key query. We remark that by a standard hybrid argument, the proof can be easily extended to the more general case where the adversary makes $\text{poly}(k)$ number of (non-adaptive and adaptive) key queries.

Now, note that the only difference between H_4 and H_5 is the manner in which the secret key SK_f for the key query f is computed: in experiment H_4 , SK_f is a differing-inputs obfuscation of

\mathcal{H}_f , while in H_5 , SK_f is a differing-inputs obfuscation of \mathcal{H}'_f . It follows that if there exists a PPT adversary \mathcal{A} that distinguishes between the outputs of H_4 and H_5 with non-negligible probability, then we can construct a PPT adversary \mathcal{A}' that distinguishes between $\text{diO}(\mathcal{H}_f)$ and $\text{diO}(\mathcal{H}'_f)$ with non-negligible probability. Then, it follows from Definition 9 that for such an adversary \mathcal{A}' , there exists a PPT extractor algorithm E that on input $(\mathcal{H}_f, \mathcal{H}'_f)$ outputs an input value z^* such that $\mathcal{H}_f(z^*) \neq \mathcal{H}'_f(z^*)$. We will use E to contradict the simulation-soundness property of the NIZK argument system (CRSGen, Prove, Verify).

Towards that end, let $z^* = (\text{CT}_1^*, \dots, \text{CT}_n^*)$, where for every $i \in [n]$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$. Without loss of generality, we assume that every proof string π_i^* is *accepting*. This is because otherwise from the definition of \mathcal{H}_f and \mathcal{H}'_f , we have that $\mathcal{H}_f(z^*) \neq \mathcal{H}'_f(z^*)$. We make the following claim about the input z^* .

Claim 49. Let $\{\hat{\text{CT}}_{1,j}, \dots, \hat{\text{CT}}_{n,j}\}_{j=1}^q$ denote the challenge ciphertexts in experiments H_3 and H_4 , where every $\hat{\text{CT}}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$. Then, there exists $i \in [n]$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ in z^* such that one of the following two cases holds:

Case 1: If $i \in I$, then $c_{i,1}^*$ and $c_{i,2}^*$ are encryptions of different messages, and for every $j \in [q]$, either $c_{i,1}^* \neq \hat{c}_1^{i,j}$ or $c_{i,2}^* \neq \hat{c}_2^{i,j}$.

Case 2: If $i \in N \setminus I$, then for every $j \in [q]$, either $c_{i,1}^* \neq \hat{c}_1^{i,j}$ or $c_{i,2}^* \neq \hat{c}_2^{i,j}$.

Proof. Suppose that the claim is false. That is, the input $z^* = (\text{CT}_1^*, \dots, \text{CT}_n^*)$ output by E is such that:

Condition A: For every $i \in I$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ is such that:

1. Either there exists $j_i \in [q]$ such that $c_{i,1}^* = \hat{c}_1^{i,j_i}$ and $c_{i,2}^* = \hat{c}_2^{i,j_i}$, or
2. $c_{i,1}^*$ and $c_{i,2}^*$ are encryptions of the *same* message. Let x'_i denote this message.

Condition B: For every $i \in N \setminus I$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$, there exists $j_i \in [q]$ such that $c_{i,1}^* = \hat{c}_1^{i,j_i}$ and $c_{i,2}^* = \hat{c}_2^{i,j_i}$.

Let us now inspect the outputs of \mathcal{H}_f and \mathcal{H}'_f on the input z^* . We have that:

$$\begin{aligned}\mathcal{H}_f(z^*) &= f\left(\left\langle \left\{ \text{PKE.Dec}(\text{sk}_1, \hat{c}_1^{i,j_i}) \right\}_{i \in N \setminus I}, \left\{ \text{PKE.Dec}(\text{sk}_1, c_{i,1}^*) \right\}_{i \in I} \right\rangle\right), \\ \mathcal{H}'_f(z^*) &= f\left(\left\langle \left\{ \text{PKE.Dec}(\text{sk}_2, \hat{c}_2^{i,j_i}) \right\}_{i \in N \setminus I}, \left\{ \text{PKE.Dec}(\text{sk}_2, c_{i,2}^*) \right\}_{i \in I} \right\rangle\right),\end{aligned}$$

where for $\ell \in [2]$, $\left\langle \left\{ \text{PKE.Dec}(\text{sk}_\ell, \hat{c}_\ell^{i,j_i}) \right\}_{i \in N \setminus I}, \left\{ \text{PKE.Dec}(\text{sk}_\ell, c_{i,\ell}^*) \right\}_{i \in I} \right\rangle$ denotes the “arrangement” of the values $\left\{ \text{PKE.Dec}(\text{sk}_\ell, \hat{c}_\ell^{i,j_i}) \right\}_{i \in N \setminus I}, \left\{ \text{PKE.Dec}(\text{sk}_\ell, c_{i,\ell}^*) \right\}_{i \in I}$ according to their input positions in f . Now, let $I' \subseteq I$ be such that for every $i \in I'$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ satisfies the condition A(2). Thus, for every $i \in I \setminus I'$, $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ satisfies the condition A(1). Then, we have:

$$\mathcal{H}_f(z^*) = f\left(\left\langle \left\{ x_{i,j_i}^0 \right\}_{i \in N \setminus I'}, \left\{ x'_i \right\}_{i \in I'} \right\rangle\right); \quad \mathcal{H}'_f(z^*) = f\left(\left\langle \left\{ x_{i,j_i}^1 \right\}_{i \in N \setminus I'}, \left\{ x'_i \right\}_{i \in I'} \right\rangle\right),$$

where $\vec{X}^0 = \{x_{1,j}^0, \dots, x_{n,j}^0\}_{j=1}^q$ and $\vec{X}^1 = \{x_{1,j}^1, \dots, x_{n,j}^1\}_{j=1}^q$ are the challenge messages.

Now, regardless of whether f is a non-adaptive or adaptive key query, it follows from the I-Compatibility property in the IND-security definition (see Definition 3) that $\mathcal{H}_f(z^*) = \mathcal{H}'_f(z^*)$, which is a contradiction. \square

Completing the proof of Lemma 48. Following the above claim, fix $\text{CT}_i^* = (c_{i,1}^*, c_{i,2}^*, \pi_i^*)$ in z^* to be such that either Case 1 or Case 2 holds. Let $y_i^* = (c_{i,1}^*, c_{i,2}^*, \text{pk}_1, \text{pk}_2, Z_i)$ be the statement corresponding to the proof string π_i^* . Further, let $\hat{y}_{i,j}$ be the statement corresponding to the proof string $\hat{\pi}^{i,j}$ in challenge ciphertext $\text{CT}_{i,j} = (\hat{c}_1^{i,j}, \hat{c}_2^{i,j}, \hat{\pi}^{i,j})$.

We consider Case 1 and Case 2 separately.

Case 1. Since $c_{i,1}^*$ and $c_{i,2}^*$ are encryptions of different messages, we have that the statement y_i^* is *false*. Further, since for all $j \in [q]$, either $c_{i,1}^* \neq \hat{c}_1^{i,j}$ or $c_{i,2}^* \neq \hat{c}_2^{i,j}$, we have that $y_i^* \neq \hat{y}_{i,j}$. Then, we have that the output z^* of the extractor algorithm E includes an *accepting* proof for a new, false statement y_i^* . This contradicts the simulation-soundness property of the NIZK argument system $(\text{CRSGen}, \text{Prove}, \text{Verify})$.

Case 2. Since Z_i is computed as a commitment to 1 in experiments H_3 and H_4 , it follows from the perfect binding property of Com that the statement y_i^* is *false*. Further, since for all $j \in [q]$, either $c_{i,1}^* \neq \hat{c}_1^{i,j}$ or $c_{i,2}^* \neq \hat{c}_2^{i,j}$, we have that $y_i^* \neq \hat{y}_{i,j}$. Then, we have that the output z^* of the extractor algorithm E includes an *accepting* proof for a new, false statement y_i^* . This contradicts the simulation-soundness property of the NIZK argument system $(\text{CRSGen}, \text{Prove}, \text{Verify})$. \square

Lemma 50 ($\text{H}_5 \stackrel{c}{\equiv} \text{H}_6$). *Assuming that $\text{PKE} = (\text{PKE.Setup}, \text{PKE.Enc}, \text{PKE.Dec})$ is a semantically-secure public-key encryption scheme, the outputs of experiments H_5 and H_6 are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 47. \square

Lemma 51 ($\text{H}_6 \stackrel{c}{\equiv} \text{H}_7$). *Assuming that diO is a differing-inputs obfuscator, Com is perfectly binding and $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is simulation-sound, the outputs of experiments H_6 and H_7 are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 48. \square

Lemma 52 ($\text{H}_7 \stackrel{c}{\equiv} \text{H}_8$). *Assuming that Com is a computationally hiding commitment scheme, the outputs of experiments H_7 and H_8 are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 46. \square

Lemma 53 ($\text{H}_8 \stackrel{c}{\equiv} \text{H}_9$). *Assuming that $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a NIZK argument system, the outputs of experiments H_8 and H_9 are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 45. \square

Lemma 54 ($\text{H}_9 \stackrel{c}{\equiv} \text{H}_{10}$). *Assuming that $(\text{CRSGen}, \text{Prove}, \text{Verify})$ is a NIZK argument system, the outputs of experiments H_9 and H_{10} are computationally indistinguishable.*

Proof. The proof follows in the same manner as Lemma 44. \square

D NA-SIM-secure MI-FE

NA-SIM security for multi-input FE is defined similarly to definition 4, except that now the adversary is required to make key queries *before* (as opposed to after) choosing the challenge messages. More concretely, we define (t, p, q) -NA-SIM-secure functional encryption where (as earlier) t denotes the number of encryption keys known to the adversary and q denotes the number of challenge messages per encryption key. The new parameter p denotes the total number of non-adaptive key queries by the adversary. Below, we present the formal definition.

Definition 55 (NA-SIM Security). *We say that a functional encryption scheme \mathcal{FE} for n -ary functions \mathcal{F} is (t, p, q) -NA-SIM-secure if for every PPT adversary $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2)$, there exists a PPT simulator $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \mathcal{S}_2)$ such that the outputs of the following two experiments are computationally indistinguishable:*

<p>Experiment $\text{REAL}_{\mathcal{A}}^{\mathcal{FE}}(1^k)$:</p> <p>$(I, \text{st}_0) \leftarrow \mathcal{A}_0(1^k)$ where $I = t$</p> <p>$(\vec{E}\mathbf{K}, \text{MSK}) \leftarrow \text{FE.Setup}(1^k)$</p> <p>$(\mathcal{M}, \text{st}_1) \leftarrow \mathcal{A}_1^{\text{FE.Keygen}(\text{MSK}, \cdot)}(\text{st}_0, \vec{E}\mathbf{K}_I)$</p> <p>$\vec{X} \leftarrow \mathcal{M}$ where $\vec{X} = \{x_{1,j}, \dots, x_{n,j}\}_{j=1}^q$</p> <p>$\text{CT}_{i,j} \leftarrow \text{FE.Enc}(\vec{E}\mathbf{K}_i, x_{i,j}) \forall i \in [n], j \in [q]$</p> <p>$\alpha \leftarrow \mathcal{A}_2(\vec{\text{CT}}, \text{st}_1)$</p> <p>Output: $(I, \mathcal{M}, \vec{X}, \{f_\ell\}, \alpha)$</p>	<p>Experiment $\text{IDEAL}_{\mathcal{S}}^{\mathcal{FE}}(1^k)$:</p> <p>$(I, \text{st}_0) \leftarrow \mathcal{S}_0(1^k)$</p> <p>$(\mathcal{M}, \text{st}_1) \leftarrow \mathcal{S}_1(\text{st}_0)$</p> <p>$\alpha \leftarrow \mathcal{S}_2^{\text{TP}(\mathcal{M}, \cdot, \cdot)}(\text{st}_1)$</p> <p>Output: $(I, \mathcal{M}, \vec{X}, \{g_\ell\}, \alpha)$</p>
---	--

where $\{f_\ell\}$ denote the queries of \mathcal{A}_1 to FE.Keygen and $\{g_\ell\}$ denote the functions appearing in the queries of \mathcal{S}_2 to TP such that $|\{f_\ell\}| = |\{g_\ell\}| = p$. The oracle $\text{TP}(\mathcal{M}, \cdot, \cdot)$ denotes the ideal world trusted party that given the message distribution \mathcal{M} , TP first samples a message vector $\vec{X} \leftarrow \mathcal{M}$, where $\vec{X} = \{x_{1,j}, \dots, x_{n,j}\}_{j=1}^q$. It accepts input queries of the form $(g, (j_1, \dots, j_n))$ and outputs $g(x_{1,j_1}, \dots, x_{n,j_n})$.

Optimally Resilient and Adaptively Secure Multi-Party Computation with Low Communication Locality

Nishanth Chandran* Wutichai Chongchitmate† Juan A. Garay‡
Shafi Goldwasser§ Rafail Ostrovsky† Vassilis Zikas¶

Abstract

Secure multi-party computation (MPC) has been thoroughly studied over the past decades. The vast majority of works assume a full communication pattern: every party exchanges messages with *all* the network participants over a complete network of point-to-point channels. This can be problematic in modern large scale networks, where the number of parties can be of the order of millions, as for example when computing on large distributed data.

Motivated by the above observation, Boyle, Goldwasser, and Tessaro [TCC 2013] recently put forward the notion of *communication locality*, namely, the total number of point-to-point channels that each party uses in the protocol, as a quality metric of MPC protocols. They proved that assuming a public-key infrastructure (PKI) and a common reference string (CRS), an MPC protocol can be constructed for computing any n -party function, with communication locality $\mathcal{O}(\log^c n)$ and round complexity $\mathcal{O}(\log^{c'} n)$, for appropriate constants c and c' . Their protocol tolerates a static (i.e., non-adaptive) adversary corrupting up to $t < (\frac{1}{3} - \epsilon)n$ parties for any given constant $0 < \epsilon < \frac{1}{3}$. These results leave open the following questions:

- (1) Can we achieve low communication locality and round complexity while tolerating *adaptive* adversaries?
- (2) Can we achieve low communication locality with *optimal resiliency* $t < n/2$?

In this work we answer both questions affirmatively. First, we consider the model from [TCC 2013], where we replace the CRS with a symmetric-key infrastructure (SKI). In this model we give a protocol with communication locality and round complexity $\text{polylog}(n)$ (as in the [TCC 2013] work) which tolerates up to $t < n/2$ *adaptive* corruptions, under a standard intractability assumption for adaptively secure protocols, namely, the existence of trapdoor permutations whose domain has invertible sampling. This is done by using the SKI to derive a sequence of random *hidden communication graphs* among players. A central new technique then shows how to use these graphs to emulate a complete network in $\text{polylog}(n)$ rounds while preserving the $\text{polylog}(n)$ locality. Second, we show how we can even remove the SKI setup assumption at the cost, however, of increasing the communication locality (but not the round complexity) by a factor of \sqrt{n} .

*Microsoft Research, India, nichandr@microsoft.com.

†UCLA, wutichai@math.ucla.edu, rafail@cs.ucla.edu.

‡Yahoo Labs, garay@yahoo-inc.com.

§MIT and The Weizmann Institute of Science, shafi@theory.csail.mit.edu.

¶ETH Zurich, Switzerland, vzikas@inf.ethz.ch.

1 Introduction

Secure multi-party computation (MPC for short) allows a set of n parties to securely compute any given function f on their private data. Ensuing the seminal works in the area [41, 26, 2, 14], the systematic study of the problem over the last decades has lead to great improvements regarding several efficiency measures, such as communication complexity (number of exchanged messages), round complexity, and computation complexity. Until recently, however, essentially all MPC results required all parties to communicate directly with each other over a complete network of point to point channels, or by having access to a broadcast channel. While this requirement may be harmless when the number of participants is small compared to the complexity of the function f , it is highly problematic in settings where the number of parties is a dominant factor¹.

Communication locality in MPC. Recently, Boyle, Goldwasser, and Tessaro [6], building on work by King *et al.* on Byzantine agreement [32, 33]², introduced a new efficiency metric called *communication locality* to address such settings. Informally, the communication locality of a protocol is the total number of different point-to-point channels that each party uses in the protocol. The protocols provided in [6] for the computation of any polynomial time function f achieve a communication locality of $\text{polylog}(n)$ assuming a public-key infrastructure (PKI), a common reference string (CRS), and the existence of a semantically secure public-key encryption and existentially unforgeable signatures. An example of a scenario where the complexity of the function may be much smaller than the number of parties, is when securely computing the output of a sublinear algorithm, which takes inputs from a small subset of $q = o(n)$ of parties. (Sublinear algorithms are particularly useful for computing statistics on large populations.) By assuming, in addition to the PKI and semantically secure public-key encryption, the existence of a multi-signature scheme [38, 37], a (certifiable) fully homomorphic encryption (FHE) [7, 8], and simulation-sound adaptive non-interactive zero-knowledge (NIZK) [4, 23], the authors also obtain a protocol for computing sublinear functions, which communicates $\mathcal{O}((\kappa + n) \cdot \text{polylog}(n))$ -bit messages³ and terminates in $\text{polylog}(n) + \mathcal{O}(q)$ rounds.

The solution of [6], however, has two major limitations:

- (1) It cannot tolerate an *adaptive* adversary who may choose the parties to corrupt on the fly during the protocol execution; it only tolerates a static adversary who decides on the faulty parties prior to the protocol execution.
- (2) It achieves a sub-optimal resiliency of $t < (1/3 - \epsilon)n$ corrupted parties, for any given constant $0 < \epsilon < 1/3$, whereas traditional MPC protocols in the computational setting (without the low communication locality requirement) can tolerate up to $t < n/2$ corruptions.

Our results. In this paper, we first show that by replacing the CRS with a slightly different setup assumption, namely, a *symmetric-key infrastructure (SKI)* [21] where every pair of participants shares a uniformly random key that is unknown to other participants, we can overcome both of the above limitations. Specifically, we construct **adaptively secure** MPC protocols with communication locality $\text{polylog}(n)$ tolerating any $t < n/2$ corruptions. (As mentioned above, this is the optimal number of corruptions that can be tolerated, even in the complete communication setting without the extra requirement of communication locality [26, 15].) Looking ahead, we will show

¹ Interestingly, recent implementation results report remarkable performance of the state-of-the-art solutions for small instances of the problem such as three-party computation [5] or in a lab environment when broadcast is assumed for free (e.g., [3, 36, 16, 17, 19, 30]).

²[32, 33] in fact achieve “almost-everywhere” Byzantine agreement [22], which does not guarantee that all honest players will receive an output (see “Other related work” below).

³ κ is the security parameter.

how the SKI can be interpreted as a special type of random initial communication graph which dictates which pairs of players can send point-to-point messages to each other to start with. The graph is shared but “hidden:” each player will only know the restricted subset of $\text{polylog}(n)$ players it can send messages to and receive messages from.⁴

Next, we show that we can remove the additional SKI assumption at the cost of increasing the communication locality by a factor of \sqrt{n} . Both our constructions assume the existence of a family of trapdoor permutations which has a *reversed domain sampler* [18, 25]. This is the weakest known general assumption which is sufficient for *non-committing encryption* [11, 18], and thus for adaptively secure MPC over non-private channels. Such families are known to exist under standard number-theoretic assumptions such as the hardness of the decisional Diffie-Hellmann problem (DDH) or the RSA assumption [18].

We remark that in order to circumvent the shortcomings in [6] we need to develop new and quite different techniques, as the limitations to sub-optimal resiliency and non-adaptive adversaries seem to be inherent in their approach. This can be seen as follows. In [6], the parties elect n *input committees* $\mathcal{C}_1, \dots, \mathcal{C}_n$, as well as one “supreme” committee \mathcal{C} —all of size $\text{polylog}(n)$ —in a way that ensures that (with high probability) at least a $2/3$ fraction of the parties in each committee are honest. Each protocol message of party p_i is then secret-shared to committee \mathcal{C}_i , which re-shares it to the parties of the supreme committee \mathcal{C} . Subsequently, the members of \mathcal{C} compute the output of the given function on the shared inputs and return it to the users (by sharing it to the input committees, which then reconstruct to their associated input parties). All sharings are private and robust so long as the adversary does not corrupt more than $1/3$ of a committee members.

Clearly, the above cannot work if the adversary is allowed to adaptively corrupt parties depending on his view of the election process. Such an adversary might choose to corrupt more than a $1/3$ fraction of the parties in some committee⁵ and thus violate the privacy of the protocol. Furthermore, even for a static adversary, the above approach cannot yield an optimally resilient (i.e., $t < n/2$) protocol, as an adversary who non-adaptively corrupts $\lceil n/2 \rceil - 1$ of the parties has a noticeable probability of corrupting $1/3$ (or even $1/2$) of the parties in some committee.

Interestingly, we note that under the additional assumptions of FHE and multi-signatures, [6] obtains better communication complexity for computing sublinear algorithms than directly applying our approach. Improving the communication complexity of our protocols is an enthralling direction for future research.

Other related work. Our result should be contrasted with the work of Dani *et al.* [20], which provides MPC in the information-theoretic setting assuming perfectly private communication channels with communication complexity of $O(\sqrt{n})$, but only offers security against a static adversary and $t < n/3$ corruptions. For the problem of Byzantine agreement (BA), King and Saia [31] show how to construct a protocol that is secure against adaptive corruptions, and where the communication complexity of every party is $\tilde{O}(n)$. This leads to a BA protocol with $\tilde{O}(n)$ communication locality; however, their protocol only tolerates $t < (\frac{1}{3} - \epsilon)n$ corruptions (and is specific to Byzantine agreement).

Another related body of work is on conducting Byzantine agreement and MPC when players are not connected via a point-to-point network but rather via a sparse, public network. This has been studied both in the context of BA [22, 40, 12, 13] and of MPC [24, 32, 33]. These results inevitably only achieve the so called *almost-everywhere* versions of the problems, as the protocols “give up” a number $x = \omega(1)$ of honest parties (and provide no guarantees for them). The interested reader

⁴In fact, one may alternatively state our setup as having the players share an initial hidden random graph, and our result as a reduction from this setup.

⁵Recall that the adversary has a linear corruption “budget” $t < (1/3 - \epsilon)n$ and the committees are of size $\text{polylog}(n)$.

may refer to Appendix A for a short survey of the corresponding literature.

1.1 Overview of our results and techniques

In this paper we establish the feasibility of secure multiparty computation with low (i.e., $\text{polylog}(n)$) communication locality both for static and for adaptive adversaries corrupting any $t < n/2$ parties. Our constructions assume a PKI and a symmetric-key infrastructure (SKI—see details below). Furthermore, our protocols have $\text{polylog}(n)$ round complexity. In more detail, we show the following:

Theorem 1. *Assuming a PKI, an SKI, and trapdoor permutations with a reversed domain sampler, there exists an MPC protocol secure against an adaptive adversary corrupting up to $t < n/2$ parties and satisfying the following properties with overwhelming probability:*

- (Polylogarithmic communication locality) *Every party communicates with at most $\mathcal{O}(\log^{1+\epsilon} n)$ other parties, for some constant $\epsilon > 0$.*
- (Polylogarithmic round complexity) *The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds, for some constant $\epsilon' > 0$.*

Since we wish to obtain MPC with guaranteed output delivery for all honest players, our bound on $t < \frac{n}{2}$ is optimal. Furthermore, if we do not wish to “give up” any party in the protocol, then the best communication locality that one can hope to attain is $\omega(\log n)^6$, and hence our protocols are near optimal in terms of communication locality as well.

Next, we show that we can completely get rid of the SKI setup (while still guaranteeing adaptive security) at the cost of increasing the communication locality (but not the round complexity). That is, we show:

Theorem 2. *Assuming a PKI and trapdoor permutations with a reversed domain sampler, there exists an MPC protocol secure against an adaptive adversary corrupting up to $t < n/2$ parties and satisfying the following conditions with overwhelming probability:*

- *Every party communicates with at most $\mathcal{O}(\sqrt{n} \log^{1+\epsilon} n)$ other parties, for some constant $\epsilon > 0$.*
- *The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds for some constant $\epsilon' > 0$.*

In the remainder of this section we summarize our main techniques and provide a high-level overview of our MPC construction. Before we do that, we describe our model in a bit more detail. All parties are connected via a complete network of point-to-point channels. For simplicity, we assume that the channels are secure; however, as we assume a public-key infrastructure (PKI), these channels can be implemented by encryption and authentication [26]. Furthermore, we assume *synchronous* communication, i.e., our protocols proceed in rounds where messages sent in any round are delivered by the end of the round. An adversary can adaptively corrupt $t < n/2$ parties and cannot observe whether or not two honest parties communicated. In addition, our construction assumes a *symmetric-key infrastructure* (denoted SKI), where every pair (i, j) of parties shares a uniformly random key $\text{sk}_{i,j} \in \{0, 1\}^\kappa$ for some security parameter κ . Note that there does not seem to be a direct way of getting rid of the SKI assumption without increasing the communication locality, as the direct approach of using the PKI for fair exchange would require (at least) a round where every party communicates with all other parties to exchange the pairwise keys. Removing the SKI assumption without increasing the locality is an intriguing open problem.

SKI as a hidden graph setup. Central to our results is a novel way of interpreting/transforming a symmetric key-infrastructure into a special type of setup, which we refer to as *hidden-graph setup* (HG).

⁶If a party communicates with only $\mathcal{O}(\log n)$ parties in the protocol, then an adversary can simply guess these $\mathcal{O}(\log n)$ parties (with non-negligible probability) and corrupt them, thereby isolating this honest party.

Let $G = (V, E)$ be an undirected graph, where $V = [n]$ is the vertex set and E is the set of edges in G . In slight abuse of notation, we also use E to denote the adjacency matrix of G , i.e., $E(i, j) = E(j, i) = 1$ if there is an edge in G connecting vertices i and j ; otherwise $E(i, j) = E(j, i) = 0$. We let $G(n, p) = (V, E)$ denote the Erdős-Rényi random graph on n vertices where for every $i, j \in V$, $\Pr[(i, j) \in E] = p$. We refer to such a graph as a p -random graph.

We say that the parties in $[n]$ hold a *hidden p -random graph setup* (p -HG)⁷ if, after sampling $G = G(n, p)$, every party $i \in [n]$ is given his corresponding row $E(i, j)$ for $j \in [n]$ and no other information on E . Note that instead of the naïve encoding which would require n bits (i.e., give each party the full vector corresponding to his row in E), we can simply give each party i a vector $\Gamma(i)$ which includes the parties i communicates with over the bilateral secure channel. Thus if party i communicates with q parties, his p -HG setup will be of size $q \log(n)$.⁸

We now show how such a HG can be efficiently (and locally) computed from a SKI: Recall that in an SKI every pair of parties i and j is given a uniformly random key $\mathbf{sk}_{i,j}$. We use this key as a seed to a pseudo-random function (PRF). Parties i and j will use the PRF (keyed with $\mathbf{sk}_{i,j}$) to (locally) compute the random coins needed to sample (i, j) for the graph G ; i.e., i and j will use the output of the PRF as coins in a sampling algorithm which picks a bit b to be 1 with probability p . If $b = 1$, then i and j will communicate with each other directly in the protocol and (i, j) will be an edge in the communication graph G . The security of the PRF ensures that the bit b computed as above is distributed indistinguishably from the output of the sampling algorithm on uniformly random coins. Without loss of generality, we will henceforth assume that the PRF keys that parties share can be used to sample as many random graphs as needed.

Our adaptively secure construction will make use of several ($\text{polylog}(n)$ -many) independent HG's. A sequence of ℓ -many HG's that is indistinguishable from a sequence of ℓ independent p -HG's can be generated as above, by querying the PRF on distinct (fixed) inputs.

Overview of our construction. At the heart of our construction lies a protocol for reliable message transmission (RMT) in this communication-constrained setting. Such a protocol allows a sender i to reliably send a message to a receiver j . Note that as we assume a completely connected network, a trivial way of implementing RMT would be for party i to use the point-to-point channel he shares with each $j \in [n]$. However, our goal is to achieve RMT where each party utilizes only a polylogarithmic number of its direct point-to-point channels. Clearly, in such a setting we cannot allow the adversary to know the neighbors of an honest party $i \in [n]$ as this would enable the adversary to “cut-off” (i.e., isolate) party i from the rest of the parties by corrupting all of its neighbors.

This is where the hidden-graph setup comes in handy: Every party will only exchange messages with its neighbors in this hidden graph and ignore all other interfaces.⁹ As we show, an adversary who corrupts up to any constant fraction $q < 1$ of parties cannot make the length of the shortest honest path between any two honest parties to be greater than $\log^{\epsilon'}(n)$, for some $\epsilon' > 0$, except with negligible probability. In particular, we show that if G' denotes the graph that is obtained by deleting from G all parties/nodes that such an adversary corrupts, then with overwhelming probability, every two nodes in G' (i.e., every two honest parties) are connected (in G') by a path of length at most $\log^{\epsilon'} n$. Thus, parties can achieve RMT by simply “flooding” the network; i.e., party i will simply send message m , signed under its signing key, to all its neighbors; then, for $\log^{\epsilon'}(n)$

⁷Throughout this paper we only consider $p = \frac{\log^{1+\epsilon}(n)}{n}$ for some $\epsilon > 0$. Whenever ϵ is clear from the context we might omit p and just refer to the setup as a “(hidden) random graph setup.”

⁸In our setting $q = \text{polylog}(n)$ with overwhelming probability, thus, we get that a hidden graph setup is also of size $\text{polylog}(n)$.

⁹Note that the adversary might try to send messages to honest parties using all the corrupted parties. However, the honest parties will ignore messages from all parties that are not their neighbors in their hidden graphs.

rounds, all parties in every round, will simply forward (the first validly signed) message that they receive to all its neighbors. Since i and j are connected by a path of length $N = \log^{\epsilon'} n$ in G' , then after N rounds, j will receive at least one copy of m that is signed under i 's signing key and hence will reliably receive the message m . Observe that the above RMT protocol tolerates any constant fraction $q < 1$ of corruptions (i.e., up to $t \leq qn$ corrupted parties) and requires a standard PKI for digital signatures (in addition to the HG). We assume standard digital signatures secure against chosen-plaintext attacks. Further, since the message is guaranteed to reach all honest parties within N rounds, the above RMT protocol can be used to have a message sent to *all* honest parties.¹⁰

Unfortunately, the above approach only works for a static adversary. The reason is that, while corrupting parties (even adaptively) and learning their setup, does not reveal anything about the hidden graph (other than the neighbors of corrupted parties themselves), the protocol itself might reveal whether or not $(i, j) \in E$ for honest parties $i, j \in [n]$. For example, if an adversarial party i sends a message to another adversarial party j , and j receives this message in 3 rounds, then it must be the case that there exists a path of length 3 between i and j . One might think that we can get around this problem by simply having i encrypt the message under j 's public key; this, however, is completely useless in the case when j is corrupted. Another idea might be to have i delay sending its message; however, this too is useless when i is corrupted.¹¹ As a result, constructing an RMT protocol for the adaptive-corruption case ends up being much more challenging than in the static case.

The high-level idea behind the protocol for the adaptive case is to sample a new Erdős-Rényi random graph $G = G(n, p)$, with $p = \frac{\log^{\epsilon} n}{n}$, at *every round* of the protocol. As long as the total number of rounds of the protocol is polylogarithmic, so will be the total number of point-to-point channels that an honest party uses (since in each round, every honest party might speak to at most $\text{polylog}(n)$ —potentially new—neighbors). The intuition for choosing a different HG for each round is that any corruptions made by the adversary before round i are independent of the graph selected in round i and hence this would be equivalent to the static adversary case. However, now proving that honest parties can communicate reliably (and that there exists a path of bounded length between any two honest parties) is delicate, constituting the crux of our technical result.

Having RMT, the next step is to design the MPC protocol. Recall that our goal is a protocol with full security (i.e., including fairness) an optimal resiliency (i.e., tolerating $t < n/2$ corruptions) [15, 26]. One idea to achieve this is as follows: Since we have already established RMT between any two honest parties, we can invoke any known MPC protocol Π secure for $t < n/2$ assuming authenticated channels, over the virtual network induced by RMT. Whenever party i is instructed in Π to send a message m to party j , we invoke RMT for this purpose. This approach would give an MPC protocol tolerating up to $t < n/2$ corruptions, but does work generically (for any protocol Π) in combination with our simulated communication channels.

To see why, observe that in our adaptively secure protocol, an increase of the round complexity implies the same (asymptotic) increase of the honest parties' communication locality. Indeed, since using our RMT, every party communicates with $\mathcal{O}(\log^{\epsilon} n)$ (potentially new) parties in every round $1 \leq \ell \leq D$, we can only afford to run a protocol that runs in $\log^{c'} n$ number of rounds for some $c' > 0$. Thus, in order for the above idea to work we need an adaptive MPC protocol over point-to-point authenticated channels which terminates in $\text{polylog}(n)$ rounds. Such a protocol can be obtained by taking any constant-round MPC protocol that utilizes a point-to-point network of secure channels and a broadcast channel (e.g., the protocol in [1]), and modifying it as follows: (1)

¹⁰Note, however, that if the sender is corrupted, there is no guarantee that the message is sent consistently.

¹¹Note that we want to use RMT for *every* pair of parties; thus, the adversary might use information on the HG learned in an execution of RMT with a corrupted sender and/or receiver to attack another RMT with honest sender and receiver.

transmission over the point-to-point secure channels are emulated by calls to our RMT protocol where the message is encrypted using non-committing encryption, and (2) calls to the broadcast channel are emulated by a (randomized, authenticated) broadcast protocol which terminates in $\text{polylog}(n)$ rounds (cf. the protocol in [29]).

Remark 1 (Static security). Our primary goal in this paper is adaptive security. However, in the static security setting our approach yields a protocol with $\text{polylog}(n)$ locality which relies only on semantically secure public-key encryption and existentially unforgeable signatures (as in [6]). The protocol tolerates an optimal number of $t < n/2$ corruptions and assumes a PKI and a (single) hidden graph setup¹²(instead of the PKI and CRS assumed in [6]).

Finally, we show (Section 5) how to avoid the SKI assumption, at the expense of an increased communication locality (but not round complexity)—cf. Theorem 2. In a nutshell, the parties will compute some kind of alternate random graph setup by having each party *locally* decide which of his n point-to-point channels he will use; a channel between two (honest) parties $i, j \in [n]$ is then used only if both parties choose it. By adequately setting the probability of the honest parties' decisions, the resulting communication graph will include an Erdős-Rényi graph which will allow us to use our ideas from the SKI-based construction, with a guaranteed $\mathcal{O}(\sqrt{n} \log^\delta n)$ communication locality, for some constant $\delta > 0$.

2 Model, Definitions and Building Blocks

As already mentioned earlier, we assume all parties share a public-key infrastructure (PKI) as well as a symmetric-key infrastructure (SKI). In other words, every party has a public-key, secret-key pair (for a digital signature scheme); every party $i \in [n]$ receives party j 's public-key (for all $j \in [n]$). In addition, every pair of parties $i, j \in [n]$ share a secret key $\text{sk}_{i,j}$. Parties are connected by a fully connected *synchronous* network; however, in our constructions every party will only communicate with $\text{polylog}(n)$ other parties.

We allow up to $t < \frac{n}{2}$ of the parties to be *adaptively* corrupted by a *rushing* adversary (meaning that the adversary is allowed to corrupt parties dynamically during the protocol execution and depending on his view, and that the adversary is able to postpone the sending of any given round's messages until after he receives the messages from the honest parties, resp.).

We consider the standard simulation-based notion of security for multiparty protocols via the real/ideal world paradigm. In other words (and informally), we require that for every probabilistic-polynomial time adversary \mathcal{A} (that corrupts t of the parties) in a real-world execution of the protocol, there exists a corresponding PPT adversary \mathcal{S} in the ideal world who can simulate the output of \mathcal{A} given only access to the ideal world where \mathcal{S} only learns the output of the evaluated function. We prove our results for standalone security. We refer the reader to [9, 10] for further details on this notion of security for multiparty computation. Throughout, we assume that $n > \kappa$, the security parameter.

Our constructions rely on the standard intractability assumption for adaptively secure multiparty protocols, namely, the existence of a family of trapdoor permutations with a reversed domain sampler [18, 25]. Informally, these are trapdoor permutations with an extra property that there exists an algorithm (the reversed domain sampler) which given an input and output can reconstruct (sample) the corresponding random bits used by the permutation function. This assumption is sufficient for all the primitives used in this paper, namely: Pseudo-random functions (PRFs) [28],

¹²Note that, instead of an SKI, a single copy of our hidden graph can be represented as $\text{polylog}(n)$ bits held by each party corresponding to the vector of the indices of its neighbours.

existentially unforgeable signatures (assuming a PKI setup) [28], constant-round non-committing encryption (informally, this is encryption which transforms an authenticated channel into a secure one in the presence of an adaptive adversary [18]), and constant-round adaptively secure MPC over a point-to-point network with (authenticated) broadcast [1] (see below).

Definition 3 ([39, 34]). A protocol for parties $\mathcal{P} = P_1, \dots, P_n$, where a distinguished player (called the dealer) $P^* \in \mathcal{P}$ holds an initial input m , is a *broadcast protocol tolerating t malicious parties* if the following conditions hold for any adversary controlling at most t parties:

- **Agreement:** All honest parties output the same value v .
- **Validity:** If the dealer is honest, then $v = m$.

Broadcast protocols that assume a public-key infrastructure are usually termed authenticated.

We also make use of the following fact about expected-constant-round broadcast and Byzantine agreement protocols, implicit in [29].

Theorem 4 ([29]). *Assuming a PKI, there exists a protocol Π_{BC} which achieves broadcast with overwhelming probability against $t < n/2$ adaptive corruptions, running for $\log^{1+c}(n)$ rounds on a complete network, for some constant $c > 0$.*

3 Reliable Communication in the Locality Model

In this section we prove our results for Reliable Message Transmission (RMT) between every pair of honest parties in our communication-constrained setting, assuming a standard PKI (for digital signatures) as well as an SKI, as defined above. The constructions in this section tolerate any constant fraction of corrupted parties than what is required for fully secure MPC; that is, we only assume that the number of corrupted parties is $t \leq qn$, for constant $q < 1$ (arbitrarily close to 1).

3.1 Static security

We first show an RMT protocol that is secure against static corruptions. This will illustrate some of the ideas that are needed for our adaptively secure construction.

Setup phase. Recall that we work in a model in which parties share a public-key as well as a symmetric-key infrastructure. That is, in the setup phase, party i receives a private key sk_i for a signature scheme, and every party j receives the public key vk_i corresponding to sk_i , for all $i \in [n]$. The SKI allows for a hidden p -random graph setup (p -HG), with $p = \frac{\log^{1+\epsilon} n}{n}$ (for appropriately chosen $\epsilon > 0$), as explained above. Note that, because in this section we assume only a single shared hidden graph, it is sufficient (in fact equivalent) that the keys in the SKI are one-bit long.

Construction idea. The hidden graph setup ensures that the adversary does not get to know whether party i communicates with party j , unless he corrupts one of them. We show that given such a p -HG, an adversary who (non-adaptively) corrupts any constant fraction q of the parties cannot isolate any of the honest parties. In fact, we show a much stronger property for the graph G' formed by removing (in the hidden graph) $t = qn$ corrupted nodes; namely, that with overwhelming probability (in n), every pair (i, j) of honest parties is connected by a path of length at most $N = \log^{\epsilon'}(n)$, for some $\epsilon' > 0$ which depends only on ϵ . Note that since parties start with a PKI, we only require that honest parties $i, j \in [n]$ are connected by a path of length $N = \log^{\epsilon'}(n)$, for some $\epsilon' > 0$ in graph G' . Parties can then achieve RMT by simply “flooding” the network; i.e., party i will simply send message m , signed under its signing key, to all its neighbors. Next, each party in every round simply forwards the (first validly signed) message that it receives to all of

its neighbors. A formal description of the non-adaptively secure protocol for a sender i to reliably send a message m to a receiver j , denoted by $\text{RMT}_{i,j}(m)$, is as follows. (Let $\Gamma(i)$ denote party i 's neighbors in G .)

Protocol $\text{RMT}_{i,j}(m)$

1. Round 1: Party i sends $(m, \text{sig}_{\text{sk}_i}(m))$ to all nodes in $\Gamma(i)$.
2. For each round $\rho = 2, \dots, \log^{\epsilon'}(n)$:
 - For every party $k \in [n] \setminus \{i, j\}$: If a message (m, σ) , where σ is party i 's valid signature on m , was received *for the first time* from some of its neighbors, i.e., some node in $\Gamma(i)$, in the previous round, then party k sends (m, σ) to all its neighbors and halts. (If multiple validly signed pairs were received in that round for the first time, then take the first one in a lexicographic order.)
 - For receiver j : If a message (m, σ) , where σ is party i 's valid signature on m , is received *for the first time* from some node in $\Gamma(j)$ then output m and halt. (If multiple validly signed pairs are received in that round for the first time, then take the first one in a lexicographic order.)

The security of protocol $\text{RMT}_{i,j}(m)$ (stated in Theorem 7) can be argued as follows: If i and j are connected by a path of length N in G' , then after N rounds j will receive at least one copy of m that is signed under i 's signing key, and hence will reliably receive the message m . Thus we simply need to argue that the above holds for some $N = \text{polylog}(n)$. To this direction, we first prove the following lemma, which implies RMT between i and j for all honest $i, j \in [n]$.

Lemma 5. *Let $G = (V, E)$ be a hidden p -random graph, and let \mathcal{A} be an adversary who non-adaptively chooses a set of parties to corrupt and by doing so learns all their neighbors in G . Denote by $U \subseteq V$ the set of corrupted nodes, and by G' the subgraph on $V \setminus U$ resulting from erasing all nodes in U . If for some constant $q < 1$, $|U| \leq qn$ and $p = \frac{d}{n} = \frac{\log^{1+\epsilon} n}{n}$, then, for any constant $0 < k < \frac{1-q}{2}$, G' is an expander graph with edge expansion kd .*

Proof. Since each pair of vertices in G' is still connected with probability p independently of U , G' is a random graph $G((1-q)n, p)$. Let $n' = (1-q)n$ and $0 < k < \frac{1-q}{2}$. Then, for each $S \subseteq V' = V \setminus U$, $|S| = r \leq \frac{n'}{2}$, we have

$$e_{G'}(S, \bar{S}) = \sum_{v \in S, v' \in \bar{S}} X_{v,v'},$$

where $X_{v,v'}$ is the indicator whether there exists an edge between v and v' . Then

$$\mathbb{E}[e_{G'}(S, \bar{S})] = \sum_{v \in S, v' \in \bar{S}} \mathbb{E}[X_{v,v'}] = |S||\bar{S}|p = r(n' - r)p.$$

By the Chernoff bound,

$$\Pr[e_{G'}(S, \bar{S}) < kd|S|] \leq e^{-\left(1 - \frac{kn}{n' - r}\right)^2 r(n' - r)p} = \left(e^{-\frac{\left(1 - \frac{kn}{n' - r}\right)^2 (n' - r)}{2n}}\right)^{rd} = \left(e^{-\frac{\left(\frac{n' - r}{n} - k\right)^2}{2 \cdot \frac{n' - r}{n}}}\right)^{rd}.$$

Since $0 < r < \frac{n'}{2}$, we have

$$\frac{1 - q}{2} = \frac{n'}{2n} \leq \frac{n' - r}{n} \leq \frac{n'}{n} = 1 - q < 1.$$

Thus,

$$\frac{\left(\frac{n'-r}{n} - k\right)^2}{2 \cdot \frac{n'-r}{n}} \geq \frac{1}{2} \cdot \left(\frac{1-q}{2} - k\right)^2 = c > 0.$$

For $d = \log^{1+\epsilon} n$, we have

$$\Pr[e_{G'}(S, \bar{S}) < kd|S|] \leq (e^{-c})^{rd} = \left(\frac{1}{n^{c' \log^\epsilon n}}\right)^r,$$

and by the union bound, the probability that $e_{G'}(S, \bar{S}) < kd|S|$ for some subset S , $|S| \leq |V|/2$ is bounded by

$$\begin{aligned} \sum_{r=1}^{\frac{n'}{2}} \sum_{S, |S|=r} \Pr[e_{G'}(S, \bar{S}) < kd|S|] &\leq \sum_{r=1}^{\frac{n'}{2}} \binom{n'}{r} \left(\frac{1}{n^{c' \log^\epsilon n}}\right)^r \\ &\leq \sum_{r=1}^{\frac{n'}{2}} n^r \left(\frac{1}{n^{c' \log^\epsilon n}}\right)^r \\ &= \sum_{r=1}^{\frac{n'}{2}} \left(\frac{1}{n^{c' \log^\epsilon n-1}}\right)^r \\ &< \frac{\frac{1}{n^{c' \log^\epsilon n-1}}}{1 - \frac{1}{n^{c' \log^\epsilon n-1}}} = \lambda(n), \end{aligned}$$

where $\lambda(n)$ represents a function that is negligible in n . Therefore, G' is an expander with edge expansion kd with overwhelming probability. \square

The next corollary follows immediately from Lemma 5, by using the fact that an expander graph as above has polylogarithmic diameter except with negligible probability. We make use of the following intuitive terminology: for a given graph $G = ([n], E)$ we say that two parties i and j in $[n]$ are G -connected by an honest path of length ℓ if there exists a sequence of connected nodes $\text{PATH}(i, j)$ from i to j in G such that for every node $k \in \text{PATH}(i, j)$, node k is honest, and $|\text{PATH}(i, j)| = \ell$.

Corollary 6. *Let $\epsilon > 0$, $p = \frac{\log^{1+\epsilon} n}{n}$, and G be a hidden p -random graph. For any adversary who (non-adaptively) corrupts at most $t = qn$ parties, the following holds except with negligible (in n) probability: there exists some $\epsilon' > 0$ which depends only on ϵ such that any two honest parties are G -connected by an honest path of length at most $\log^{\epsilon'}(n)$.*

The security of protocol $\text{RMT}_{i,j}(m)$ follows now easily from the above corollary, as no matter how the (static) adversary chooses the corrupted parties he cannot increase the diameter of the graph defined by the honest parties and the hidden graph setup to more than $\text{polylog}(n)$.

Theorem 7. *Let $0 < q < 1$, and $T \subset [n]$ be the set of (non-adaptively) corrupted parties, $|T| = t \leq qn$. Assuming a PKI and an SKI, then $\text{RMT}_{i,j}$ is a secure RMT protocol between any two honest nodes $i, j \in [n] \setminus T$ satisfying the following two conditions with overwhelming probability:*

1. *Every party communicates with at most $\mathcal{O}(\log^{1+\epsilon} n)$ other parties;*
2. *the protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds, for some $\epsilon' > 0$.*

Proof. Since Lemma 5 shows that any message sent by an honest i will reach every honest j within $\mathcal{O}(\log^{\epsilon'}(n))$ rounds, it follows from the unforgeability property of the signature scheme that j will always accept the message sent by honest i . Hence, the above protocol is a secure RMT protocol. The communication locality of the protocol follows from the degree of $G = G(n, p)$ which is $\mathcal{O}(\log^{1+\epsilon} n)$, except with negligible probability. \square

Parallel composition of RMT. In our MPC construction, we will require all nodes to execute their respective RMT protocols in parallel (simultaneously). That is, let $m_{i,j}$ be the message that node i wishes to send to j via the RMT protocol, denoted $\text{RMT}_{i,j}(m_{i,j})$ as above. Now, let $\text{RMT}_{\text{all}}(\mathbf{m})$ denote the protocol executed by all parties when $\text{RMT}_{i,j}(m_{i,j})$ for all $i, j \in [n]$ are executed in parallel. (That is, in round k of $\text{RMT}_{\text{all}}(\mathbf{m})$, all parties execute the k^{th} round of protocol $\text{RMT}_{i,j}(m_{i,j})$, for all $i, j \in [n]$). $\text{RMT}_{\text{all}}(\cdot)$ is composed of n^2 individual RMT protocols. We have the following corollary.

Corollary 8. *For all honest $i, j \in [n]$, $\text{RMT}_{\text{all}}(\mathbf{m})$ is a reliable message transmission protocol for sending $m_{i,j}$ from i to j , satisfying the following properties:*

1. *Every party communicates with at most $\mathcal{O}(\log^{1+\epsilon} n)$ other parties in the protocol.*
2. *The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds for some $\epsilon' > 0$.*

Proof. From Lemma 5 we have that any message sent by any honest i will reach every honest j within $\mathcal{O}(\log^{\epsilon'} n)$ rounds. Hence, from this and the unforgeability of the underlying signature scheme, it follows by a standard hybrid argument that every honest j will always accept the message sent by any honest i at the end of $\text{RMT}_{\text{all}}(\mathbf{m})$. Furthermore, note that the protocol's round complexity is equal to the maximum round complexity of its components, which equals $\mathcal{O}(\log^{\epsilon'} n)$. Further, note that the communication locality of every party in $\text{RMT}_{\text{all}}(\mathbf{m})$ is equal to the communication locality of the party in $\text{RMT}_{i,j}(m_{i,j})$, for any $i, j \in [n]$. Hence, the corollary follows. \square

3.2 Adaptively secure RMT

As discussed in the Section 1.1 the above proof technique fails against adaptive adversaries. Informally, the issue is that an adversary can use the round in which a corrupted party/relayer receives a message to deduce information on the communication graph (see Section 1.1 for more details and a concrete example). In this section we describe an RMT protocol that is secure against such an *adaptive* adversary. The idea is have the parties use a different, independent communication graph for each round in the transmission scheme. As long as the transmission scheme does not have more than $\text{polylog}(n)$ rounds and in each round, every party communicates with at most $\text{polylog}(n)$ (additional) parties, the overall locality will be $\text{polylog}(n)$.

The main challenge in the above idea is to prove that in this dynamically updated communication graph, the message will reach each recipient through an honest path in at most $\text{polylog}(n)$ rounds. Proving this constitute the main technical contribution of our work. The (adaptively secure) RMT protocol **AdRMT** is similar to the protocol in the static case, except that in round ρ parties forward messages received in the previous round to their neighbours in the communication graph G_ρ . We first describe the corresponding setup that it requires.

Setup phase. As in the static case, the parties share both a PKI and an SKI. The SKI will be used here in the same spirit, except that instead of generating one Erdős-Rényi graph, $G = G(n, p)$ with $p = \frac{\log^{\epsilon} n}{n}$, it will be used to generate D such graphs, denoted $\mathcal{G} = (G_1, \dots, G_D)$. These graphs can be sampled using the same PRF key $\text{sk}_{i,j}$ that parties i and j share. As before, every node only

knows its own neighbors, and when the adversary corrupts a node j , he only learns j 's neighbors in G_1, \dots, G_D .

The protocol is described below, followed by security statement and a high-level description of its proof. (The formal proof can be found in Appendix B.)

Protocol AdRMT _{i,j} (m)

1. Round 1: Party i sends $(m, \text{sig}_{\text{sk}_i}(m))$ to all its neighbors in graph G_1 .
2. For each round $\rho = 2, \dots, \log^{\epsilon'}(n)$:
 - For every party $k \in [n] \setminus \{i, j\}$: If a message (m, σ) , where σ is party i 's valid signature on m was received *for the first time* from some of its neighbours in $G_{\rho-1}$ in the previous round, then party k sends (m, σ) to all its neighbors in graph G_ρ and halts. (If multiple validly signed pairs were received in that round for the first time, then take the first one in a lexicographic order.)
 - For receiver j : If a message (m, σ) , where σ is party i 's valid signature on m is received *for the first time* from some of party j 's neighbours in G_ρ , then output m and halt. (If more than one validly signed pair is received in that round for the first time, then take the first one in a lexicographic order.)

Theorem 9. *Let $T \subset [n]$ be the set of adaptively corrupted parties, $|T| = t \leq qn$, for any constant $0 < q < 1$. Assuming a PKI and an SKI, protocol AdRMT _{i,j} (m) is a secure RMT protocol between any two honest nodes $i, j \in [n] \setminus T$, satisfying the following two properties with overwhelming probability:*

1. *Every party communicates with at most $\mathcal{O}(\log^{1+\epsilon} n)$ other parties.*
2. *The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds, for some $\epsilon' > 0$.*

Proof idea. As in the static case, we show that there exists a path of length at most $\mathcal{O}(\log^{\epsilon'}(n))$ between any two honest nodes $i, j \in [n]$ when we consider the collection of communication graphs \mathcal{G} that selects graph G_i as the communication graph in hop i . We prove this in three steps:

First, we prove that at every step of the protocol, even if an adversary corrupts a constant fraction of the nodes in the random graph, the honest neighbors of any set S of size $\leq \frac{n}{d}$ that are not in S , will be at least of size $kd|S|$, for some appropriate constant k (except with negligible probability). More concretely, in Appendix B we prove the following lemma, where we let $\epsilon > 0$, $0 < q < 1$ be constants, $d = \log^{1+\epsilon} n$, $p = \frac{d}{n} = \frac{\log^{1+\epsilon} n}{n}$, and $D = \mathcal{O}(\log n)$.

Lemma 10. *Let $G = G(n, p)$ be graph on $V = [n]$, and $U \subseteq V$, $|U| \leq qn$, chosen adaptively while only learning edges connecting to U . Let G' be the induced subgraph on $V' = V \setminus U$. Then, for any constant $0 < k < \frac{1-q}{2}$, there exists a constant $c > 0$ such that, for sufficiently large n and for any $S \subseteq V'$ with $|S| = r \leq \frac{n}{d} = \frac{1}{p}$, the set of all neighbors of S that are not in S , $\Gamma(S)$, has size at least $kd|S|$ except with negligible probability $P_r = \left(\frac{1}{n^c \log^\epsilon n}\right)^r$.*

Next, via an application of Hoeffding's inequality (see Lemma 16 in Appendix B,) we prove that as long as the adversarial parties are chosen independently of the random neighbors chosen by any party, a constant fraction of the party's neighbors will be honest, except with negligible probability (as long as the adversarial set is of size at most qn for some constant $0 < q < 1$). Thus we get the following.

Lemma 11. *Let $V = [n]$ and $C \subseteq V$, $|C| = m$, be a subset chosen uniformly at random. Let $0 < q < 1$ be a constant and $U \subseteq V$, $|U| = qn$, be a subset chosen independently of C . Then, for all $0 < \delta < 1 - q$, $|C \setminus U| > (1 - q - \delta)m$ except with probability $e^{-2m\delta^2}$. In particular, for $m = \log^{1+\epsilon'} n$, $|C \setminus U| > \left(\frac{1-q}{2}\right)m$ except with negligible probability. Furthermore, for $q = \frac{1}{2} - \epsilon$, $|C \setminus U| > \frac{1}{2}m$ except with negligible probability.*

Finally, using Lemmas 10 and 11, we show that even when an adversary adaptively corrupts parties in every round of the protocol, as long as the parties select a random graph at each round of the protocol, there exists a path of length at most $D = \mathcal{O}(\log n)$ between any two honest nodes in $[n]$. Formally:

Lemma 12. *Let G_1, \dots, G_D be graphs on $V = [n]$ constructed independently as $G(n, p)$. Let $U_1, U_2, \dots, U_D \subseteq V$ be disjoint subsets with $U = \cup_{i=1}^D U_i$ such that $|U| = qn$ where U_j is chosen independently from G_{j+1}, \dots, G_D , but adaptively, after learning the neighbors of U_i in G_i for $i \leq j$. Let G'_i be the induced subgraph on $V_i = V \setminus (\cup_{j=1}^i U_j)$. Then, except with negligible probability, any pair of vertices $v, v' \in V' = V \setminus U$ are reachable with respect to $\mathcal{G}' = (G'_1, \dots, G'_D)$ by a path of length at most D .*

Combining these gives us our main theorem (Theorem 9). \square

Parallel composition of adaptively secure RMT. Once again, we will require all nodes $i, j \in [n]$ to execute their respective RMT protocols in parallel simultaneously. Let $\text{AdRMT}_{\text{all}}(\mathbf{m})$ denote the protocol executed by all parties when $\text{AdRMT}_{i,j}(m_{i,j})$ for all $i, j \in [n]$ are executed in parallel. That is, in round k of $\text{AdRMT}_{\text{all}}(\mathbf{m})$, all parties execute the k^{th} round of protocol $\text{AdRMT}_{i,j}(m_{i,j})$ (for all $i, j \in [n]$). Note that the graph G_k used in the k^{th} round of the protocol depends only on the round k and not on i and j ; hence, we use the same graph G_k to send all the messages of protocol $\text{AdRMT}_{\text{all}}(\mathbf{m})$. We have the following corollary:

Corollary 13. *For all honest $i, j \in [n]$, $\text{AdRMT}_{\text{all}}(\mathbf{m})$ is a reliable message transmission protocol for sending $m_{i,j}$ from i to j , satisfying the following properties:*

1. *Every party communicates with at most $\mathcal{O}(\log^{1+\epsilon} n)$ other parties in the protocol.*
2. *The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds, for some $\epsilon' > 0$.*

The proof of this corollary is similar to Corollary 8's.

4 Secure Multiparty Computation with Low Communication

We are now ready to describe our MPC protocol for securely evaluating any given (even reactive) n -party function in the communication-locality model. Our protocol is secure against $t < n/2$ adaptive corruptions. The idea behind our MPC protocol is to use a constant-round adaptively secure MPC protocol for $t < n/2$ working over point-to-point secure channels and broadcast (e.g., [1]), where those resources are emulated via our RMT protocol of Section 3.2.

We let Π_{BC} denote the authenticated broadcast protocol guaranteed by Theorem 4 (Section 2). The protocol achieves broadcast with overwhelming probability against $t < n/2$ adaptive corruptions, running for $\log^{1+c} n$ rounds on a complete network, for some constant $c > 0$. As pointed out in [29], assuming unique process and message ID's as in [35], Π_{BC} remains secure under parallel composition.

Let Π_{BC}^* denote the protocol which results by having the parties execute Π_{BC} where in each round instead of using the point-to-point channels for exchanging their messages, the parties invoke $\text{AdRMT}_{\text{all}}$ from Section 3.2. Then it follows immediately from the security of $\text{AdRMT}_{\text{all}}$

(Corollary 13) and the fact that each message transmission requires $\text{polylog}(n)$ rounds that protocol Π_{BC}^* is also a secure broadcast protocol with polylogarithmic round complexity and communication locality.

Lemma 14. *Protocol Π_{BC}^* described above achieves broadcast against $t < n/2$ adaptive corruptions and satisfies the following conditions with overwhelming probability:*

1. *Every party communicates with at most $\mathcal{O}(\log^{1+\epsilon} n)$ parties for any constant $\epsilon > 0$.*
2. *The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds for some constant $\epsilon' > 0$.*

Proof (sketch). The security of Π_{BC}^* follows directly from the security of protocols Π_{BC} and $\text{AdRMT}_{\text{all}}$. The (asymptotic) round complexity is computed as follows: for each round ℓ of Π_{BC} , protocol Π_{BC}^* executes $\text{AdRMT}_{\text{all}}$ to have the parties exchange their round ℓ messages; thus, for each round in Π_{BC} we need $\mathcal{O}(\log^{\epsilon''} n)$ rounds in Π_{BC}^* . Because Π_{BC} runs in $\mathcal{O}(\log^{\epsilon'} n)$ rounds, the total round complexity of Π_{BC}^* is $\mathcal{O}(\log^{\epsilon'+\epsilon''} n)$ rounds. We next argue the communication locality: With overwhelming probability, in each round of Π_{BC}^* , every party might communicate with at most to $\mathcal{O}(\log^{1+\epsilon} n)$ (potentially different) parties (for executing $\text{AdRMT}_{\text{all}}$). Thus, since the total number of rounds is $\mathcal{O}(\log^{\epsilon'+\epsilon''} n)$, then with overwhelming probability (by the union bound) the total number of parties that each $i \in [n]$ exchanges messages with using the point-to-point channels is $\mathcal{O}(\log^{1+\epsilon+\epsilon'+\epsilon''} n)$. \square

The next step is to construct a *secure* message transmission protocol (SMT) which will allow a sender i to securely (i.e., authentically and privately) send a message $m_{i,j}$ to a receiver j . Since we have a PKI and an adaptively secure broadcast protocol, we can use the standard reduction of secure channels to broadcast: The sender i encrypts $m_{i,j}$ under the receiver's public key and broadcasts the corresponding ciphertext $c_{i,j}$. Upon receiving $c_{i,j}$, party j decrypts it using his secret key and recovers $m_{i,j}$. However, in order for the above reduction to be secure (in a simulation-based manner) against an adaptive adversary, we need to ensure that a simulator can “open” a ciphertext to any message of its choice. This can be achieved by the use of a *non-committing encryption* scheme for computing the ciphertext $c_{i,j}$ [11]. As proved in [18] constant-round non-committing encryption can be constructed assuming the existence of families of trapdoor permutations with a reversed domain sampler. Consistently with the notation introduced in the previous section, we use $\text{AdSMT}_{i,j}$ to denote the above SMT protocol, and $\text{AdSMT}_{\text{all}}$ to denote the protocol composed of n^2 individual $\text{AdSMT}_{i,j}(m_{i,j})$ protocols (for all $i, j \in [n]$), run in parallel, where $\mathbf{m} = (m_{1,1}, m_{1,2}, \dots, m_{nn})$.

With the above tools, we have:

Theorem 1. *Assuming a PKI, an SKI, and trapdoor permutations with a reversed domain sampler, there exists a protocol for securely evaluating any given n -party function against an adaptive adversary who corrupts $t < n/2$ parties, satisfying the following two conditions with overwhelming probability:*

1. *Every party communicates with at most $\mathcal{O}(\log^{1+\epsilon} n)$ other parties, for some constant $\epsilon > 0$.*
2. *The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds, for some constant $\epsilon' > 0$.*

Proof (sketch). Let Π_{MPC} denote a constant-round MPC protocol which is secure against adaptive corruptions of up to $t < n/2$ parties, where parties communicate over a complete network of point-to-point channels and broadcast. (Such protocols are known to exist under the assumption in the theorem, e.g., [1].) Furthermore, let Π_{MPC}^* denote the protocol that results by instantiating in Π_{MPC} the calls to the secure channels and broadcast by invocations of protocols Π_{BC}^* and AdSMT , respectively. We argue that Π_{MPC}^* satisfies all the properties claimed in the theorem. The security of Π_{MPC}^* follows immediately from the security of the underlying protocol Π_{MPC} and the security of protocols

Π_{BC}^* and $\text{AdSMT}_{\text{all}}$. For the round complexity: For each round in Π_{MPC} , all message exchanges (i.e., point-to-point transmissions or broadcast calls) are exchanged in Π_{MPC}^* by appropriate (parallel) executions of protocols Π_{BC}^* and $\text{AdSMT}_{\text{all}}$, where the executions have unique round, protocol, and message IDs.¹³ Thus, for every round in Π_{MPC} we need $\mathcal{O}(\log^{\epsilon'} n)$ rounds in Π_{MPC}^* , for some given constant $\epsilon' > 0$. Because Π_{MPC} terminates in a constant number of rounds, the round complexity of Π_{MPC}^* is also $\mathcal{O}(\log^{\epsilon'} n)$. In each of these rounds, every party might communicate with at most $\mathcal{O}(\log^{1+\epsilon} n)$ (potentially different) parties, (Recall that all parallel executions of Π_{BC}^* and $\text{AdSMT}_{\text{all}}$ use the same sequence of graph setups.) Thus, the total number of parties that each $i \in [n]$ talks directly to (i.e., via its point-to-point channels) is $\mathcal{O}(\log^{1+\epsilon+\epsilon'} n)$. \square

5 Getting Rid of the SKI

In this section we show how to get rid of the symmetric-key setup assumption, at the cost, however, of increasing the communication-locality (but not the round complexity) by a factor of \sqrt{n} .

The idea for getting rid of the SKI is to have the parties compute some kind of an alternative random graph setup. This is done as follows: each party $i \in [n]$ locally decides which of his n point-to-point channels he will use; a channel between two (honest) parties $i, j \in [n]$ is then used only if both parties choose it. (This is similar in spirit to the way the work of Chandran *et al.* [13] handles “edge corruptions” in sparse networks.) By having each party decide to use each of his channels with probability $p = \frac{\log^{\epsilon} n}{\sqrt{n}}$ for some given constant $\epsilon > 1$ (and ignore all other channels) we ensure that, with overwhelming probability, each (honest) party uses at most $\mathcal{O}(\sqrt{n} \log^{\delta} n)$ of its point-to-point channels for some constant $\delta > 0$. Furthermore, each edge between two honest parties i and j is chosen with probability $p' = p^2 = \frac{\log^{2\epsilon} n}{n}$, thus the resulting communication graph will include Erdős-Rényi graph $G(n, p')$ which will allow us to use our ideas from the previous sections. Note however, that as the adversarial nodes might choose to communicate with all their neighbors, the communication locality is no longer guaranteed to be $\mathcal{O}(\log^{\epsilon} n)$; notwithstanding, it is guaranteed to be $\mathcal{O}(\sqrt{n} \log^{\delta} n)$ with overwhelming probability.

RMT protocol. We now describe a reliable message transmission protocol which tolerates up to $t < qn$ adaptive corruptions, for any given constant $q < 1$. Our protocol (and proof) are similar to the corresponding protocol from Section 3.2, with the only difference being that the parties choose their neighbors in a setup procedure as above instead of sampling them by use of a PRF keyed with their SKI-keys.

¹³Recall that the ID’s are needed to ensure security of Π_{BC}^* under parallel composition [35].

Protocol AdRMT _{i,j} ^{noSKI}(m)

1. Round 1 (Computing the setup): The parties execute the following code for every $(i, j, \rho) \in [n] \times [n] \times [\log^{\epsilon'} n]$ in parallel (where $\epsilon' > 1$ is a given constant):
 - Party i samples a bit $b_{i,j}^\rho$ where $b_{i,j}^\rho = 1$ with probability $p = \frac{\log^{\epsilon'} n}{\sqrt{n}}$ for some given constant $\epsilon > 1$; and $b_{i,j}^\rho = 0$ otherwise.
 - If $b_{i,j}^\rho = 0$ for all $\rho \in [\log^{\epsilon'} n]$, then party i ignores all messages on the point-to-point channel between i and j .
 - If $b_{i,j}^\rho = 1$ then party i sends $(b_{i,j}^\rho, \rho)$ to party j .
2. Round 2^a: For each $(i, j, \rho) \in [n] \times [n] \times [\log^{\epsilon'} n]$: If $b_{i,j}^\rho = 1$ but party i received no message (b, ρ) from party j in the previous round then i sets $b_{i,j}^\rho := 0$. For $\rho = 1, \dots, \log^{\epsilon'} n$: Party i sets $\Gamma(i)^\rho := \{j \mid b_{i,j}^\rho = 1\}$ to be the set of parties/neighbors p_i will communicate with in round ρ .
3. Round 3: Party i sends $(m, \text{sig}_{\text{sk}_i}(m))$ to parties in $\Gamma(i)^\rho$.
4. For each round $\rho = 3, \dots, \log^{\epsilon'} n$:
 - For every party $k \in [n] \setminus \{i, j\}$: If a message (m, σ) , where σ is party i 's valid signature on m was received for the first time in the previous round $\rho - 1$ from some party in $\Gamma(k)^{\rho-1}$, then party k sends (m, σ) to all parties in $\Gamma(k)^\rho$ and halts. (If multiple validly signed pairs were received in that round for the first time, then take the first one in a lexicographic order.)
 - For the receiver j : If a message (m, σ) , where σ is party i 's valid signature on m is received for the first time from some party in $\Gamma(j)^\rho$, then output m and halt. (If more than one validly signed pair is received in that round for the first time, then take the first one in a lexicographic order.)

^aThis round is redundant and could be executed at the beginning of the following round. Nonetheless, we include it here because it simplifies the description and it does not affect the (asymptotic) round complexity argument.

Theorem 15. Let $T \subset [n]$ be the set of adaptively corrupted parties, $|T| = t \leq qn$, for any constant $0 < q < 1$. Assuming a PKI, protocol AdRMT _{i,j} ^{noSKI}(m) is a secure RMT protocol between any two honest nodes $i, j \in [n] \setminus T$, satisfying the following two properties with overwhelming probability:

1. Every party communicates with at most $\mathcal{O}(\sqrt{n} \log^{1+\delta} n)$ other parties, for some constant $\delta > 0$.
2. The protocol terminates after $\mathcal{O}(\log^{\epsilon''} n)$ rounds, for some constant $\epsilon'' > 0$.

Proof (sketch). The proof that the round complexity is $\mathcal{O}(\log^{\epsilon''} n)$ follows along the lines of Theorem 9, because for each pair of honest $i, j \in [n]$ and each $\rho = 1, \dots, \log^{\epsilon'} n$ the set $\Gamma(i)^{\rho-1}$ is distributed as in an Erdős-Rényi graph, $G = G(n, p')$ with $p' = \frac{\log^{2\epsilon'} n}{n}$. The communication locality is argued as follows: It follows from a Chernoff bound that in each round $\rho \in \{1, \dots, \log^{\epsilon'} n\}$ each party talks to at most $L = \mathcal{O}(\sqrt{n} \log^{1+c} n)$ neighbors, for some constant $c > 0$, except with negligible probability. Thus with overwhelming probability the total number of neighbors that i chooses in all $\log^{\epsilon'} n + 2$ rounds is $\mathcal{O}(\sqrt{n} \log^{1+c+\epsilon'} n)$. Because honest parties ignore all parties that they do not choose as neighbors the total number of parties that party i communicates with is at most $\mathcal{O}(\sqrt{n} \log^{1+c+\epsilon'} n)$. \square

Given Theorem 15, an MPC protocol with the desired communication-locality and round complexity can be obtained by replacing in protocol Π_{MPC}^* all invocations of AdRMT _{i,j} with invocations of AdRMT _{i,j} ^{noSKI}. The proof is similar to the proof of Theorem 1.

Theorem 2. Assuming a PKI and the existence of trapdoor permutations with a reversed domain sampler, there exists a protocol for securely evaluating any given n -party function against an adap-

tive adversary who corrupts $t < n/2$ parties. The protocol satisfies the following properties with overwhelming probability:

1. Every party communicates with at most $\mathcal{O}(\sqrt{n} \log^{1+\epsilon} n)$ other parties, for some constant $\epsilon > 0$.
2. The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds, for some constant $\epsilon' > 0$.

References

- [1] Donald Beaver, Silvio Micali, and Phillip Rogaway. The round complexity of secure protocols (extended abstract). In *22nd Annual ACM Symposium on Theory of Computing*, pages 503–513, Baltimore, Maryland, USA, May 14–16, 1990. ACM Press.
- [2] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.
- [3] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *Advances in Cryptology – EUROCRYPT 2011*, volume 6632 of *Lecture Notes in Computer Science*, pages 169–188, Tallinn, Estonia, May 15–19, 2011. Springer, Berlin, Germany.
- [4] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 103–112, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.
- [5] Dan Bogdanov, Sven Laur, and Jan Willemson. Sharemind: A framework for fast privacy-preserving computations. In Sushil Jajodia and Javier López, editors, *ESORICS 2008: 13th European Symposium on Research in Computer Security*, volume 5283 of *Lecture Notes in Computer Science*, pages 192–206, Málaga, Spain, October 6–8, 2008. Springer, Berlin, Germany.
- [6] Elette Boyle, Shafi Goldwasser, and Stefano Tessaro. Communication locality in secure multiparty computation - how to run sublinear algorithms in a distributed setting. In *TCC*, pages 356–376, 2013.
- [7] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325, Cambridge, Massachusetts, USA, January 8–10, 2012. Association for Computing Machinery.
- [8] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, California, USA, October 22–25, 2011. IEEE Computer Society Press.
- [9] Ran Canetti. Security and composition of cryptographic protocols: A tutorial. Cryptology ePrint Archive, Report 2006/465, 2006. <http://eprint.iacr.org/2006/465>.
- [10] Ran Canetti. Security and composition of cryptographic protocols: a tutorial (part i). *SIGACT News*, 37(3):67–92, 2006.

- [11] Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th Annual ACM Symposium on Theory of Computing*, pages 639–648, Philadelphia, Pennsylvania, USA, May 22–24, 1996. ACM Press.
- [12] Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. Improved fault tolerance and secure computation on sparse networks. In *ICALP (2)*, pages 249–260, 2010.
- [13] Nishanth Chandran, Juan A. Garay, and Rafail Ostrovsky. Edge fault tolerance on sparse networks. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 452–463, 2012.
- [14] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *20th Annual ACM Symposium on Theory of Computing*, pages 11–19, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.
- [15] Richard Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In Juris Hartmanis, editor, *STOC*, pages 364–369. ACM, 1986.
- [16] Ivan Damgård, Marcel Keller, Enrique Larraia, Christian Miles, and Nigel P. Smart. Implementing AES via an actively/covertly secure dishonest-majority MPC protocol. In Ivan Visconti and Roberto De Prisco, editors, *SCN 12: 8th International Conference on Security in Communication Networks*, volume 7485 of *Lecture Notes in Computer Science*, pages 241–263, Amalfi, Italy, September 5–7, 2012. Springer, Berlin, Germany.
- [17] Ivan Damgård, Marcel Keller, Enrique Larraia, Valerio Pastro, Peter Scholl, and Nigel P. Smart. Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *ESORICS 2013: 18th European Symposium on Research in Computer Security*, volume 8134 of *Lecture Notes in Computer Science*, pages 1–18, Egham, UK, September 9–13, 2013. Springer, Berlin, Germany.
- [18] Ivan Damgård and Jesper Buus Nielsen. Improved non-committing encryption schemes based on a general complexity assumption. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 432–450, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Berlin, Germany.
- [19] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 643–662, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Berlin, Germany.
- [20] Varsha Dani, Valerie King, Mahnush Movahedi, and Jared Saia. Brief announcement: breaking the $O(nm)$ bit barrier, secure multiparty computation with a static adversary. In *ACM Symposium on Principles of Distributed Computing, PODC '12, Funchal, Madeira, Portugal, July 16-18, 2012*, pages 227–228, 2012.
- [21] Yevgeniy Dodis, Jonathan Katz, Adam Smith, and Shabsi Walfish. Composability and on-line deniability of authentication. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 146–162. Springer, Berlin, Germany, March 15–17, 2009.
- [22] Cynthia Dwork, David Peleg, Nicholas Pippenger, and Eli Upfal. Fault tolerance in networks of bounded degree (preliminary version). In Juris Hartmanis, editor, *STOC*, pages 370–379. ACM, 1986.

- [23] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st Annual Symposium on Foundations of Computer Science*, pages 308–317, St. Louis, Missouri, October 22–24, 1990. IEEE Computer Society Press.
- [24] Juan A. Garay and Rafail Ostrovsky. Almost-everywhere secure computation. In *EUROCRYPT*, pages 307–323, 2008.
- [25] Oded Goldreich. Basing non-interactive zero-knowledge on (enhanced) trapdoor permutations: The state of the art. In Oded Goldreich, editor, *Studies in Complexity and Cryptography*, pages 406–421. Springer-Verlag, Berlin, Heidelberg, 2011.
- [26] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th Annual ACM Symposium on Theory of Computing*, pages 218–229, New York City, New York, USA, May 25–27, 1987. ACM Press.
- [27] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, March 1963.
- [28] Russell Impagliazzo and Michael Luby. One-way functions are essential for complexity based cryptography (extended abstract). In *30th Annual Symposium on Foundations of Computer Science*, pages 230–235, Research Triangle Park, North Carolina, October 30 – November 1, 1989. IEEE Computer Society Press.
- [29] Jonathan Katz and Chiu-Yuen Koo. On expected constant-round protocols for byzantine agreement. In Cynthia Dwork, editor, *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 445–462, Santa Barbara, CA, USA, August 20–24, 2006. Springer, Berlin, Germany.
- [30] Marcel Keller, Peter Scholl, and Nigel P. Smart. An architecture for practical actively secure MPC with dishonest majority. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 13: 20th Conference on Computer and Communications Security*, pages 549–560, Berlin, Germany, November 4–8, 2013. ACM Press.
- [31] Valerie King and Jared Saia. Breaking the $o(n^2)$ bit barrier: scalable byzantine agreement with an adaptive adversary. In *Proceedings of the 29th Annual ACM Symposium on Principles of Distributed Computing, PODC 2010, Zurich, Switzerland, July 25-28, 2010*, pages 420–429, 2010.
- [32] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Scalable leader election. In *SODA*, pages 990–999, 2006.
- [33] Valerie King, Jared Saia, Vishal Sanwalani, and Erik Vee. Towards secure and scalable computation in peer-to-peer networks. In *FOCS*, pages 87–98, 2006.
- [34] Leslie Lamport, Robert E. Shostak, and Marshall C. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
- [35] Yehuda Lindell, Anna Lysyanskaya, and Tal Rabin. On the composition of authenticated byzantine agreement. In *34th Annual ACM Symposium on Theory of Computing*, pages 514–523, Montréal, Québec, Canada, May 19–21, 2002. ACM Press.

- [36] Yehuda Lindell, Eli Oxman, and Benny Pinkas. The IPS compiler: Optimizations, variants and concrete efficiency. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 259–276, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Berlin, Germany.
- [37] Steve Lu, Rafail Ostrovsky, Amit Sahai, Hovav Shacham, and Brent Waters. Sequential aggregate signatures and multisignatures without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology – EUROCRYPT 2006*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Berlin, Germany.
- [38] Silvio Micali, Kazuo Ohta, and Leonid Reyzin. Accountable-subgroup multisignatures: Extended abstract. In *ACM CCS 01: 8th Conference on Computer and Communications Security*, pages 245–254, Philadelphia, PA, USA, November 5–8, 2001. ACM Press.
- [39] Marshall C. Pease, Robert E. Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.
- [40] Eli Upfal. Tolerating linear number of faults in networks of bounded degree. In *PODC*, pages 83–89, 1992.
- [41] Andrew Chi-Chih Yao. How to generate and exchange secrets (extended abstract). In *27th Annual Symposium on Foundations of Computer Science*, pages 162–167, Toronto, Ontario, Canada, October 27–29, 1986. IEEE Computer Society Press.

A Almost-Everywhere Protocols

Prior to [6], two lines of works have studied the problem of constructing protocols for BA/MPC in which every party communicates with only a few other parties in the protocol:

Protocols on incomplete networks. The vast majority of results for BA and MPC protocols work in a model in which, every party involved in the protocol, shares a reliable and secure channel with every other party. In large scale networks, such as the internet, such an assumption is infeasible and this leads us to the question of whether one can construct BA and MPC protocols in which every party communicates only with a few other parties. For the case of BA, the first work to consider this problem was that of Dwork, Peleg, Pippenger, and Upfal [22], who constructed various graphs of specific degrees on which one could run BA protocols. For example, they construct a graph G of degree $d = \mathcal{O}(n^\epsilon)$, for any constant $0 < \epsilon < 1$, along with a BA protocol in which every party in the protocol communicates only with its neighbors in G . Such a protocol could tolerate $t = \alpha n$ corrupt parties (for some constant $\alpha < \frac{1}{3}$). As another example, they also construct a graph of constant degree, along with a BA protocol, that could tolerate $t = \mathcal{O}(\frac{n}{\log n})$ corrupt parties.

Now, since in their model, the communication graph is fixed and chosen prior to the adversary corrupting parties, one cannot hope to achieve BA among all honest parties (as an adversary could always corrupt just the neighbors of some honest party, thereby isolating it). Hence Dwork *et al.* introduce and achieve the notion of almost-everywhere (a.e.) BA that unavoidably “gives up” x honest nodes (and provides no guarantees for these honest nodes). In their protocols, $x = \mathcal{O}(t)$. Somewhat surprisingly, Upfal [40], constructed graphs of constant degree, along with a BA protocol, that could tolerate $t = \alpha n$ corrupt parties (for some constant $\alpha < \frac{1}{3}$); unfortunately, the running time of Upfal’s algorithm is exponential (in n). To date, the best bounds known in this model are

due to Chandran, Garay, and Ostrovsky [12], who achieve a polynomial time BA protocol with parameters $d = \mathcal{O}(\log^c n)$ (for some constant $c > 1$), $t = \alpha n$ and $x = \mathcal{O}(\frac{n}{\log n})$.

For the case of secure computation, Garay and Ostrovsky [24], introduced the notion of almost-everywhere MPC (similar in spirit to a.e. BA) and showed how to take any a.e. BA protocol and convert it into an a.e. MPC with the same (asymptotic) parameters. We remark that all the above protocols provide information-theoretic security against an adaptive, computationally-unbounded, adversary that can corrupt parties at any time during (or after) the protocol.

Protocols on complete networks. One could also consider a model in which parties are connected by a complete network, but only talk to a few other parties during the protocol. Once again this gives rise to protocols with low communication locality. Indeed, the works of King, Saia, Sanwalani, and Vee [32, 33] consider this model and construct protocols for the task of leader election as well as a.e. Byzantine agreement in which every party has a communication locality of $\mathcal{O}(\log^c n)$ (for some constant $c > 1$). In fact, King *et al.* show a stronger result and construct protocols in which every party only sends $\mathcal{O}(\log^c n)$ bits in the entire protocol. However, unlike the works on incomplete networks, the works of King *et al.* [32, 33] only consider the case of *static* adversaries (i.e., they are secure only against an adversary that corrupts $t = \alpha n$ of the parties, for some constant $\alpha < \frac{1}{3}$, before the start of the protocol). These works also provide information-theoretic security.

B Proof of Theorem 9 (Adaptively secure RMT)

Hoeffding's Lemma

Lemma 16. (Hoeffding's Inequality [27]) *Let $S = \{x_1, \dots, x_N\}$ be a finite set of real numbers with $a = \min_i x_i$ and $b = \max_i x_i$. Let X_1, \dots, X_n be a random sample drawn from S without replacement.*

Let $\bar{X} = \frac{\sum_{i=1}^n X_i}{n}$ and $\mu = \frac{\sum_{i=1}^N x_i}{N} = \mathbb{E}[X_j]$. Then for all $\delta > 0$, $\Pr[\bar{X} - \mu \geq \delta] \leq e^{-\frac{2n\delta^2}{(b-a)^2}}$.

Theorem 9. *Let $T \subset [n]$ be the set of adaptively corrupted parties, $|T| = t \leq qn$, for any constant $0 < q < 1$. Assuming a PKI and an SKI, protocol $\text{AdRMT}_{i,j}(m)$ is a secure RMT protocol between any two honest nodes $i, j \in [n] \setminus T$, satisfying the following two properties with overwhelming probability:*

1. *Every party communicates with at most $\mathcal{O}(\log^{1+\epsilon} n)$ other parties.*
2. *The protocol terminates after $\mathcal{O}(\log^{\epsilon'} n)$ rounds, for some $\epsilon' > 0$.*

Proof. In the following, we provide details on the proof sketched in Section 3.2. In particular we show that there exists a path of length at most $\mathcal{O}(\log^{\epsilon'}(n))$ between any two honest nodes $i, j \in [n]$ when we consider the collection of communication graphs \mathcal{G} that selects graph G_i as the communication graph in hop i . The proof follows then easily similarly to the proof of Theorem 7.

As sketched in Section 3.2, to prove the above statement we proceed in three steps:

1. First, we shall prove in Lemma 10 that at every step of the protocol, even if an adversary corrupts a constant fraction of the nodes in the random graph, the honest neighbors of any set S of size $\leq \frac{n}{d}$ that are not in S , will be at least of size $kd|S|$, for some appropriate constant k (except with negligible probability).

2. Next, via an application of Hoeffding's inequality, we will prove in Lemma 11 that as long as the adversarial parties are chosen independently of the random neighbors chosen by any party, a constant fraction of the party's neighbors will be honest, except with negligible probability (as long as the adversarial set is of size at most qn for some constant $0 < q < 1$).
3. Finally, using Lemmas 10 and 11, we will show in Lemma 12 that even when an adversary adaptively corrupts parties in every round of the protocol, as long as the parties select a random graph at each round of the protocol, there exists a path of length at most $D = \mathcal{O}(\log n)$ between any two honest nodes in $[n]$.

Combining these will give us our main theorem (Theorem 9).

Step 1. To begin, let $\epsilon > 0, 0 < q < 1$ be constants. Let $d = \log^{1+\epsilon} n$, $p = \frac{d}{n} = \frac{\log^{1+\epsilon} n}{n}$ and $D = \mathcal{O}(\log n)$.

Lemma 10. *Let $G = G(n, p)$ be graph on $V = [n]$, and $U \subseteq V$, $|U| \leq qn$, chosen adaptively while only learning edges connecting to U . Let G' be the induced subgraph on $V' = V \setminus U$. Then, for any constant $0 < k < \frac{1-q}{2}$, there exists a constant $c > 0$ such that, for sufficiently large n and for any $S \subseteq V'$ with $|S| = r \leq \frac{n}{d} = \frac{1}{p}$, the set of all neighbors of S that are not in S , $\Gamma(S)$, has size at least $kd|S|$ except with negligible probability $P_r = \left(\frac{1}{n^c \log^\epsilon n}\right)^r$.*

Proof. Let $0 < k < \frac{1-q}{2}$ and $S \subseteq V'$ with $|S| = r \leq \frac{n}{d} = \frac{1}{p}$. Denote $n' = |V'| \geq (1-q)n$. Since each pair of vertices in G' is connected with probability p independently of U and other edges, G' is a random graph $G(n', p)$.

For each $v \in V' \setminus S$, let X_v be the indicator of whether $v \in \Gamma(S) = \Gamma_{G'}(S)$. Then

$$\Pr[X_v = 0] = \Pr[\text{no edge between } v \text{ and any vertex in } S] = (1-p)^r.$$

Since $rp < 1$,

$$\mathbb{E}[X_v] = \Pr[X_v = 1] = 1 - (1-p)^r = rp - \binom{r}{2}p^2 + \dots > \frac{rp}{2}.$$

Then

$$\mathbb{E}[|\Gamma(S)|] = \mathbb{E}\left[\sum_{v \notin S} X_v\right] > \frac{(n'-r)rp}{2}.$$

Since the X_v 's are independent, by the Chernoff Bound,

$$\Pr[|\Gamma(S)| \leq (1-\delta)\frac{(n'-r)rp}{2}] \leq \Pr[|\Gamma(S)| \leq (1-\delta)\mathbb{E}[|\Gamma(S)|]] \leq e^{-\frac{\delta^2 \mathbb{E}[|\Gamma(S)|]}{2}} \leq e^{-\frac{\delta^2 (n'-r)rp}{4}}.$$

Now let $\delta = 1 - \frac{2kn}{n'-r}$. Since $r \leq \frac{n}{d}$, we have

$$(1-q) - \frac{1}{d} \leq \frac{n'-r}{n} \leq \frac{n'}{n} < 1.$$

Let n be large enough such that $d = \log^{1+\epsilon} n > \frac{2}{1-q-2k}$. Then

$$c_0 = \frac{1}{16} \cdot ((1-q) - 2k)^2 \leq \frac{1}{4} \cdot \left((1-q) - \frac{1}{d} - 2k\right)^2 \leq \frac{\left(\frac{n'-r}{n} - 2k\right)^2}{4 \cdot \frac{n'-r}{n}}.$$

Thus,

$$\Pr[|\Gamma(S)| \leq kdr] \leq e^{-\frac{\left(1 - \frac{2kn}{n'-r}\right)^2 (n'-r)rp}{4}} = \left(e^{-\frac{\left(\frac{n'-r}{n} - 2k\right)^2}{4 \cdot \frac{n'-r}{n}}} \right)^{dr} \leq \left(\frac{1}{e^{c_0}} \right)^{dr} = \left(\frac{1}{n^{c \log^\epsilon n}} \right)^r.$$

where $c = c_0 \log e$. \square

We now proceed to show that as long as parties pick a fresh random graph in every round of the protocol, there exists at least one path of length at most D between any two honest parties $i, j \in [n]$ that does not include any corrupted party. We formally define this through the notion of *reachability with respect to \mathcal{G}* .

Definition 17. Let $\mathcal{G} = (G_1, \dots, G_D)$ be an ordered collection of graphs on subsets (V_1, \dots, V_D) of V . A pair of vertices $v \in V_1, v' \in V_l$ are *reachable with respect to \mathcal{G} by a path of length l* if there exist $v_1, \dots, v_{l-1} \in V$, such that $(v_{i-1}, v_i) \in E(G_i)$, for $i = 1, \dots, l$, where $v_0 = v$ and $v_l = v'$. We denote $N_l(v) = N_l^{\mathcal{G}}(v) \subseteq V_l$ the subset of all vertices that are reachable from v with respect to \mathcal{G} with a path of length l .

Step 2. We first make use of Hoeffding's lemma (stated in Appendix B) in order to prove a lemma that we will use. We show:

Lemma 11. Let $V = [n]$ and $C \subseteq V$, $|C| = m$, be a subset chosen uniformly at random. Let $0 < q < 1$ be a constant and $U \subseteq V$, $|U| = qn$, be a subset chosen independently of C . Then, for all $0 < \delta < 1 - q$, $|C \setminus U| > (1 - q - \delta)m$ except with probability $e^{-2m\delta^2}$. In particular, for $m = \log^{1+\epsilon'} n$, $|C \setminus U| > \left(\frac{1-q}{2}\right)m$ except with negligible probability. Furthermore, for $q = \frac{1}{2} - \epsilon$, $|C \setminus U| > \frac{1}{2}m$ except with negligible probability.

Proof. Let $S = \{x_1, \dots, x_n\}$ where $x_i = 1$ if $i \in U$, 0 otherwise. Then $a = \min_i x_i = 0$, $b = \max_i x_i = 1$ and $\mu = \frac{\sum_{i=1}^n x_i}{n} = q$. For each $i = 1, \dots, m$, let X_i be the indicator of whether each element of C is in U . Then X_i is a random sample drawn from S without replacement, and $|C \cap U| = \sum_{i=1}^m X_i = m\bar{X}$. By Hoeffding's Inequality,

$$\Pr[|C \cap U| \geq (q + \epsilon)m] = \Pr[\bar{X} - \mu \geq \delta] \leq e^{-2m\delta^2}.$$

Therefore, except with probability $e^{-2m\delta^2}$, $|C \setminus U| = m - |C \cap U| > (1 - q - \delta)m$.

Now let $m = \log^{1+\epsilon'} n$ and $\delta = \frac{1-q}{2}$. We have that $|C \setminus U| > \left(\frac{1-q}{2}\right)m$ except with probability

$$e^{-2\left(\frac{1-q}{2}\right)^2 \log^{1+\epsilon'} n} = \frac{1}{n^{c \log^{\epsilon'} n}},$$

where $c = \frac{1}{2}(1-q)^2 \log e$.

Finally, let $q = \frac{1}{2} - \epsilon$ and $\delta = \epsilon$. We have that $|C \setminus U| > \left(1 - \left(\frac{1}{2} - \epsilon\right) - \epsilon\right)m = \frac{1}{2}m$ except with probability $\frac{1}{n^{c' \log^{\epsilon'} n}}$, where $c' = 2\epsilon^2 \log e$. \square

Remark 2. Note that this proof allows U to be chosen according to any distribution. The result holds as long as C is chosen uniformly. In particular, we may allow U to be chosen adaptively.

Step 3. We now show:

Lemma 12. *Let G_1, \dots, G_D be graphs on $V = [n]$ constructed independently as $G(n, p)$. Let $U_1, U_2, \dots, U_D \subseteq V$ be disjoint subsets with $U = \cup_{j=1}^D U_j$ such that $|U| = qn$ where U_j is chosen independently from G_{j+1}, \dots, G_D , but adaptively, after learning the neighbors of U_i in G_i for $i \leq j$. Let G'_i be the induced subgraph on $V_i = V \setminus (\cup_{j=1}^i U_j)$. Then, except with negligible probability, any pair of vertices $v, v' \in V' = V \setminus U$ are reachable with respect to $\mathcal{G}' = (G'_1, \dots, G'_D)$ by a path of length at most D .*

Proof. For each $v \in V'$, we will show that, except with negligible probability, there exists $l = l(v) \leq D$ such that $V' \subseteq N_l(v) \cup N_{l+1}(v)$. Hence, by the union bound over $|V'| = (1 - q)n$ vertices, the proposition holds except with negligible probability.

Fix $v \in V'$ and choose a constant k as in Lemma 10. For each i , denote $r_i = |N_i(v) \setminus U_{i+1}|$. Note that $\Gamma_{G'_{i+1}}(N_i(v) \setminus U_{i+1}) \subseteq N_{i+1}(v)$. For i such that $r_i \leq \frac{n}{d}$, we have

$$|N_{i+1}(v)| \geq |\Gamma_{G'_{i+1}}(N_i(v) \setminus U_{i+1})| > kd|N_i(v) \setminus U_{i+1}|$$

except with probability P_{r_i} by Lemma 10.

Since U_{i+1} is chosen from V_i independently of $N_i(v)$, and $N_i(v)$ is uniform on V_i , by Lemma 11, except with negligible probability (call it P'_i),

$$|N_i(v) \setminus U_{i+1}| > \left(\frac{1-q}{2}\right) |N_i(v)|.$$

Inductively, $r_i = |N_i(v) \setminus U_{i+1}| > \left(\left(\frac{1-q}{2}\right) kd\right)^i$ and eventually greater than $\frac{n}{d}$ except with probability $\sum_{i=1}^{l_0} (P_{r_i} + P'_i)$, where l_0 is the largest integer such that $r_{l_0} \leq \frac{n}{d}$. Since $l_0 \ll D = \mathcal{O}(\log n)$ as $d^D \gg n$, this probability is negligible.

Let $n' = |V'| = (1 - q)n$. There are two possibilities for $r_{l_0+1} = |N_{l_0+1}(v) \setminus U_{l_0+2}|$: either 1) $\frac{n}{d} < r_{l_0+1} \leq \frac{n'}{2}$ or 2) $r_{l_0+1} > \frac{n'}{2}$.

Case 1: Assume that $\frac{n}{d} < r_{l_0+1} \leq \frac{n'}{2}$. Denote $r = r_{l_0+1}$ and $n_0 = |V_r| \geq n'$. Then $\frac{n}{d} = \frac{1}{p} < r \leq \frac{n'}{2} \leq \frac{n_0}{2}$. For sufficiently large n , we have $(1 - p)^{\frac{1}{p}} \approx e^{-1}$. Thus, $\mathbb{E}[|\Gamma(N_{l_0+1}(v))|] \approx (n_0 - r)(1 - e^{-rp})$. As in the proof of Lemma 10, by the Chernoff bound, we have

$$\begin{aligned} \Pr[|\Gamma(N_{l_0+1}(v) \setminus U_{l_0+2})| \leq \frac{n_0}{4}] &\leq e^{-\frac{\left(1 - \frac{n_0}{4(n_0-r)(1-e^{-rp})}\right)^2 (n_0-r)(1-e^{-rp})}{2}} \\ &\leq \left(e^{-\frac{\left(\frac{(n_0-r)(1-e^{-rp})}{n_0} - \frac{1}{4}\right)^2}{2 \cdot \frac{(n_0-r)(1-e^{-rp})}{n_0}}} \right)^{n_0} \\ &\leq \frac{1}{c'^{n_0}} \leq \frac{1}{c'^{n'}}, \end{aligned}$$

where $c' = e^{\frac{\frac{1}{2}(1-e^{-1})-\frac{1}{4}}{2}} > 1$ as $1 - e^{-1} < 1 - e^{-rp} < 1$ and $\frac{1}{2} \leq \frac{n_0-r}{n_0} < 1$. Thus, except with negligible probability,

$$r_{l_0+2} = |N_{l_0+2}(v) \setminus U_{l_0+3}| \geq \left(\frac{1-q}{2}\right) |\Gamma(N_{l_0+1}(v) \setminus U_{l_0+2})| > \left(\frac{1-q}{8}\right) n_0 \geq \left(\frac{1-q}{8}\right) n'$$

by Lemma 11. In this case, let $l = l_0 + 2$.

Case 2: $r_{l_0+1} > \frac{n'}{2}$. In this case, let $l = l_0 + 1$.

In both cases, we have $|N_l(v) \setminus U_{l+1}| = r_l > c_2 n'$ for some constant $0 < c_2 < 1$ except with negligible probability. Then, for each $v \in V' \setminus N_l(v)$, the probability that v does not connect to any vertex in $N_l(v) \setminus U_{l+1}$ is $(1-p)^{r_l} \approx e^{-r_l p} \leq \frac{1}{n^{c_3 \log^\epsilon n}}$, where $c_3 = c_2(1-q) \log e$. By the union bound, the probability that any node in $V' \setminus N_l(v)$ is not in $\Gamma(N_l(v) \setminus U_{l+1}) \subseteq N_{l+1}(v)$ is at most $\frac{1}{n^{c_3 \log^\epsilon n - 1}}$, which is negligible. Hence, except with negligible probability, $V' = N_l(v) \cup \Gamma(N_l(v)) \subseteq N_l(v) \cup N_{l+1}(v)$. Therefore, any $v' \in V'$ is reachable from v by a path of length at most D . \square

This completes the proof of Theorem 9. \square

Functional Signatures and Pseudorandom Functions

Elette Boyle
MIT

Shafi Goldwasser^{*†}
MIT and Weizmann

Ioana Ivan
MIT

October 29, 2013

Abstract

In this paper, we introduce two new cryptographic primitives: *functional digital signatures* and *functional pseudorandom functions*.

In a functional signature scheme, in addition to a master signing key that can be used to sign any message, there are *signing keys for a function f* , which allow one to sign any message in the range of f . As a special case, this implies the ability to generate keys for predicates P , which allow one to sign any message m , for which $P(m) = 1$.

We show applications of functional signatures to constructing succinct non-interactive arguments and delegation schemes. We give several general constructions for this primitive based on different computational hardness assumptions, and describe the trade-offs between them in terms of the assumptions they require and the size of the signatures.

In a functional pseudorandom function, in addition to a master secret key that can be used to evaluate the pseudorandom function F on any point in the domain, there are additional *secret keys for a function f* , which allow one to evaluate F on any y for which there exists an x such that $f(x) = y$. As a special case, this implies *pseudorandom functions with selective access*, where one can delegate the ability to evaluate the pseudorandom function on inputs y for which a predicate $P(y) = 1$ holds. We define and provide a sample construction of a functional pseudorandom function family for prefix-fixing functions.

This work appeared in part as the Master Thesis of Ioana Ivan filed May 22 at MIT. We note that independently the notion of pseudorandom functions with selective access was studied by Boneh-Waters under the name of *constrained pseudorandom functions* [BW13] and by Kiayias, Papadopoulos, Triandopoulos and Zacharias under the name *delegatable pseudorandom functions* [KPTZ13]. Subsequent to our posting of an earlier manuscript of this work, Bellare and Fuchsbaauer [BF13] and Backes, Meiser, and Schröder [BMS13] additionally posted similar results on functional signatures.

^{*}This work was supported in part by Trustworthy Computing: NSF CCF-1018064.

[†]This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

1 Introduction

We introduce new cryptographic primitives with a variety of accompanying constructions: *functional digital signatures (FDS)*, *functional pseudorandom functions (F-PRF)*, and *pseudorandom functions with selective access (PRF-SA)*.¹

Functional Signatures

In digital signature schemes, as defined by Diffie and Hellman [DH76], a signature on a message provides information which enables the receiver to verify that the message has been created by a proclaimed sender. The sender has a secret *signing key*, used in the signing process, and there is a corresponding verification key, which is public and can be used by anyone to verify that a signature is valid. Following Goldwasser, Micali and Rackoff [GMR88], the standard security requirement for signature schemes is unforgeability against chosen-message attack: an adversary that runs in probabilistic polynomial time and is allowed to request signatures for a polynomial number of messages of his choice, cannot produce a signature of any new message with non-negligible probability.

In this work, we extend the classical digital signature notion to what we call *functional signatures*. In a functional signature scheme, in addition to a *master signing key* that can be used to sign any message, there are secondary *signing keys for functions* f (called sk_f), which allow one to sign any message in the range of f . These additional keys are derived from the master signing key. The notion of security we require such a signature scheme to satisfy is that any probabilistic polynomial time (PPT) adversary, who can request signing keys for functions $f_1 \dots f_l$ of his choice, and signatures for messages m_1, \dots, m_q of his choice, can only produce a signature of a message m with non-negligible probability, if m is equal to one of the queried messages m_1, \dots, m_q , or if m is in the range of one of the queried functions $f_1 \dots f_l$.

An immediate application of a functional signature scheme is the ability to delegate the signing process from a master authority to another party. Suppose someone wants to allow their assistant to sign on their behalf only those messages with a certain tag, such as “signed by the assistant”. Let P be a predicate that outputs 1 on messages with the proper tag, and 0 on all other messages. In order to delegate the signing of this restricted set of messages, one would give the assistant a signing key for the following function:

$$f(m) := \begin{cases} m & \text{if } P(m) = 1 \\ \perp & \text{otherwise} \end{cases}.$$

P could also be a predicate that checks if the message does not contain a given phrase, or if it is related to a certain subject, or if it satisfies a more complex policy.

Another application of functional signatures is to certify that only allowable computations were performed on data. For example, imagine the setting of a digital camera that produces signed photos (i.e the original photos produced by the camera can be certified). In this case, one may want to allow photo-processing software to perform minor touch-ups of the photos, such as changing the color scale or removing red-eyes, but not allow more significant changes such as merging two photos or cropping a picture. But, how can an original photo which is slightly touched-up be distinguished from one which is the result of a major change? Functional signatures can naturally address this problem by providing the photo processing software with keys which enable it to sign only the allowable modifications of an original photograph. Generalizing, we think of a client and a server (e.g. photo-processing software), where the client provides the server with data (e.g. signed original photos, text documents, medical data) which he wants to be processed in a restricted fashion. A functional signature of the processed data provides proof of allowable processing.

Functional signatures can also be used to construct a delegation scheme. In this setting, there a client who wants to allow a more powerful server to compute a function f on inputs chosen by the client, and

¹We note that independently (and unknown to the authors) the notion of pseudorandom functions with selective access was studied by Boneh-Waters under the name of *constrained pseudorandom functions* [BW13] and by Kiayias, Papadopoulos, Triandopoulos and Zacharias under the name *delegatable pseudorandom functions* [KPTZ13]. Subsequent to our posting of an earlier manuscript of this work, [BF13] and [BMS13] have additionally posted similar results on functional signatures.

wants to be able to verify that the result returned by the server is correct. The verification process should be more efficient than for the client to compute f himself. The client can give the server a key for the function $f'(x) = (f(x)|x)$. To prove that $y = f(x)$ the prover gives the client a signature of $y|x$, which he could only have obtained if $y|x$ is in the range of f' , that is, if $y = f(x)$.

A desirable property of a functional signature scheme is *function privacy*: the signature should reveal neither the function f that the secret key used in the signing process corresponds to, nor the message m that f was applied to. In the example with the signed photos, one might not wish to reveal the original image just that the final photographs were obtained by running one of the allowed functions on some image taken with the camera.

An additional desirable property is *succinctness*: the size of the signature should only depend on the size of the output $f(m)$ and the security parameter (or just the security parameter), rather than the size of the circuit for computing f .

Functional Pseudorandomness

Pseudorandom functions, introduced by Goldreich, Goldwasser, and Micali [GGM86], are a family of indexed functions $F = \{F_s\}$ such that: (1) given the index s , F_s can be efficiently evaluated on all inputs (2) no probabilistic polynomial-time algorithm *without* s can distinguish evaluations $F_s(x_i)$ for inputs x_i 's of its choice from random values. Pseudorandom functions are useful for numerous symmetric-key cryptographic applications, including generating passwords, identify-friend-or-foe systems, and symmetric-key encryption secure against chosen ciphertext attacks. In the public-key setting, there is a construction of digital signatures from pseudorandom functions [BG89], via the following paradigm: one may publish a commitment to secret key s and henceforth be able to prove that $y = F_s(x)$ for a pair (x, y) via a non-interactive zero-knowledge (NIZK) proof.

In this work, we extend pseudorandom functions to a primitive which we call *functional pseudorandom functions* (*F-PRF*). The idea is that in addition to a master secret key (that can be used to evaluate the pseudorandom function F_s on any point in the domain), there are additional *secret keys* sk_f per function f , which allow one to evaluate F_s on any y for which there exists x such that $f(x) = y$ (i.e. $y \in \text{Range}(f)$). An immediate application of such a construct is to specify succinctly the randomness to be used by parties in a randomized distributed protocol with potentially faulty players, so as to force honest behavior. A centralized authority holds a description of an index s of a pseudorandom function F_s . One may think of this authority as providing a service which dispenses pseudorandomness (alternatively, the secret s can be shared among players in an MPC). The authority provides each party id with a secret key s_{id} which enables party id to (1) evaluate $F_s(y)$ whenever $y = "id||h"$, where h corresponds to say the public history of communication, and (2) use $F_s(y)$ as her next sequence of coins in the protocol. To prove that the appropriate randomness was used, id can utilize NIZK proofs. An interesting open question is how to achieve a *verifiable* F-PRF, where there is additional information vk_s that can be used to verify that a given pair $(x, F_s(x))$ is valid, without assuming the existence of an honestly generated common reference string, as in the NIZK setting. Note that in this example the function $f(x) = y$ is simply the function which appends the string prefix id to x . We note that there are many other ways to force the use of proper randomness in MPC protocols by dishonest parties, starting with the classical paradigm [GM82, GMW86] where parties interact to execute a "coin flip in the well" protocol forcing players to use the results of these coins, but we find the use of F-PRF appealing in its simplicity, lack of interaction and potential efficiency.

The notion of functional pseudorandom functions has many variations. One natural variant that immediately follows is *pseudorandom functions with selective access*: start with a pseudorandom function as defined in [GGM86], and add the ability to generate secondary keys sk_{P_i} (per predicate P_i) which enable computing $F_s(x)$ whenever $P_i(x) = 1$. This is a special case of F-PRF, as we can take the secret key for predicate P_i to be sk_{f_i} where $f_i(x) = x$ if $P_i(x) = 1$ and \perp otherwise. The special case of *punctured PRFs*, in which secondary keys allow computing $F_s(x)$ on all inputs except one, is similarly implied and has recently been shown to have important applications (e.g., [SW13, HSW13]). Another variant is *hierarchical pseudorandom functions*, with an additional property that parties with functional keys sk_f may also generate subordinate

keys sk_g for functions g of the form $g = f \circ f'$ (i.e., first evaluate some function f' , then evaluate f). Note that the range of such composition g is necessarily contained within the range of f .

Independent Work. A preliminary version of this work appeared in a Masters Thesis submitted on May 22, 2013. We note that independently (and unknown to the authors) the notion of pseudorandom functions with selective access was studied by Boneh-Waters under the name of *constrained pseudorandom functions* [BW13] and by Kiayias, Papadopoulos, Triandopoulos and Zacharias under the name *delegatable pseudorandom functions* [KPTZ13]. Subsequent to our posting of an earlier manuscript of this work, [BF13] and [BMS13] have additionally posted similar results on functional signatures.

1.1 Our Results on Functional Signatures and Their Applications

We provide a construction of functional signatures achieving function privacy and succinctness, assuming the existence of succinct non-interactive arguments of knowledge (SNARKS) and (standard) non-interactive zero-knowledge arguments of knowledge (NIZKAoKs) for NP languages.

As a building block, we first give a construction of a functional signature scheme that is not succinct or function private, based on a much weaker assumption: the existence of one-way functions.

Theorem 1.1 (Informal). *Based on any one-way function, there exists a functional signature scheme that supports signing keys for any function f computable by a polynomial-sized circuit. This scheme satisfies the unforgeability requirement for functional signatures, but not function privacy or succinctness.*

Overview of the construction: The master signing and verification keys for the functional signature scheme will correspond to a key pair, (msk, mvk) , in an underlying (standard) signature scheme.

To generate a signing key for a function f , we do the following. First, sample a fresh signing and verification key pair (sk', vk') in the underlying signature scheme, and sign the concatenation $f|vk'$ using msk . The signing key for f consists of this signature together with sk' . Given this signing key, a user can sign any message $m^* = f(m)$ by signing m using sk' , and outputting this signature, together with the signature of $f|vk'$ given as part of sk_f .

We then now show how to use a SNARK system, together with this initial construction, to construct a *succinct, function-private* functional signature scheme.

A SNARK system for an NP language L with corresponding relation R is an extractable proof system where the size of a proof is sublinear in the size of the witness corresponding to an instance. SNARK schemes have been constructed under various non-falsifiable assumptions. Bitansky et al. [BCCT13] construct zero-knowledge SNARKs where the length of the proof and the verifier's running time are bounded by a polynomial in the security parameter, and the *logarithm* of running time of the corresponding relation $R(x, w)$, assuming the existence of collision resistance hash functions and a knowledge of exponent assumption.² (More details are given in Section 2.3.)

Theorem 1.2 (Informal). *Assuming the existence of succinct non-interactive arguments of knowledge (SNARKs), NIZKAoK for NP languages, and a functional signature scheme that is not necessarily function-private or succinct, there exists a succinct, function-private functional signature scheme that supports signing keys for any function f computable by a polynomial-sized circuit.*

Overview of the construction: Our construction makes use of non-succinct, non-function-private functional signature scheme FS1 (which exists based on one-way functions by our construction above), and a zero-knowledge SNARK system for NP.

In the setup algorithm for our functional signature scheme, we sample a key pair (msk, mvk) for the functional signature scheme FS1, and common reference string crs for the SNARK system. We use msk as

²In [BCCT12], Bitansky et al. also show that any SNARK + NIZKAoK directly yield zero-knowledge (ZK)-SNARK with analogous parameters.

the new master signing key and (mvk, crs) as the new master verification key. The key generation algorithm is the same as in the underlying functional signature scheme FS1. To sign a message m^* using a resulting key sk_f , we generate a zero-knowledge SNARK for the following statement: $\exists \sigma$ such that σ is a valid signature of m^* under mvk in the functional signature scheme FS1. To verify the signature, we run the verification algorithm for the SNARK argument system.

Resorting to non-falsifiable assumptions, albeit strong, seems necessary to obtain succinctness for functional signatures. We show that, given a functional signature scheme with short signatures, we can construct a SNARG system.

Theorem 1.3 (Informal). *If there exists a functional signature scheme supporting keys for all polynomial-sized circuits f , that has short signatures (i.e. of size $\text{poly}(k) \cdot (|f(m)| + |m|)^{o(1)}$ for security parameter k), then there exists a SNARG scheme with preprocessing for any language $L \in \text{NP}$ with proof size $\text{poly}(k) \cdot (|w| + |x|)^{o(1)}$, where w is the witness and x is the instance.*

The main idea in the SNARG construction is for the verifier (CRS generator) to give out a single signing key sk_f for a function whose range consists of exactly those strings that are in L . Then, with sk_f , the prover will be able to sign only those messages x that are in the language L , and thus can use this (short) signature as his proof.

Gentry and Wichs showed in [GW11] that SNARG schemes with proof size $\text{poly}(k) \cdot (|w| + |x|)^{o(1)}$ cannot be obtained using black-box reductions to falsifiable assumptions. We can thus conclude that in order to obtain a functional signature scheme with signature size $\text{poly}(k) \cdot (|f(m)| + |m|)^{o(1)}$ we must either rely on non-falsifiable assumptions (as in our SNARK construction) or make use of non black-box techniques.

Finally, we can construct a scheme which satisfies unforgeability and functional privacy but not succinctness based on the weaker assumption of non-interactive zero-knowledge arguments of knowledge (NIZKAoK) for NP.

Theorem 1.4 (Informal). *Assuming the existence of non-interactive zero-knowledge arguments of knowledge (NIZKAoK) for NP, there exists a functional signature scheme that supports signing keys for any function f computable by a polynomial-sized circuit. This scheme satisfies function privacy, but not succinctness: the size of the signature is dependent on the size of f and m .*

Overview of the construction: The construction is analogous to the SNARK-based construction in the previous construction, with the SNARK system replaced with a NIZKAoK system. Namely, a signature will be a NIZKAoK for the following statement: $\exists \sigma$ such that σ is a valid signature of m^* under mvk , in an underlying non-succinct, non-function-private functional signature scheme, as before (recall such a scheme exists based on OWF). The signature size is now polynomial in the size of σ , which, if $m^* = f(m)$, and σ was generated using sk_f , is itself polynomial in the security parameter, $|m|$, and $|f|$.

1.1.1 Relation to Delegation:

Functional signatures are highly related to delegation schemes. A delegation scheme allows a client to outsource the evaluation of a function f to a server, while allowing the client to verify the correctness of the computation more efficiently than computing the function himself. We show that given *any* functional signature scheme supporting a class of functions \mathcal{F} , we can obtain a delegation scheme in the preprocessing model for functions in \mathcal{F} , with related parameters.

Theorem 1.5 (Informal). *If there exists a functional signature scheme for function class \mathcal{F} , with signature size $s(k)$, and verification time $t(k)$, then there exists a one-round delegation scheme for functions in \mathcal{F} , with server message size $s(k)$ and client verification time $t(k)$.*

Overview of the construction: The client can give the server a key $\text{sk}_{f'}$ for the function $f'(x) = (f(x)|x)$. To prove that $y = f(x)$, the prover gives the client a signature of $y|x$, which he could only have obtained if $y|x$ is in the range of f' ; that is, if $y = f(x)$. The length of a proof is equal to the length of a signature

in the functional signature scheme, $s(k)$, and the verification time for the delegation scheme is equal to the verification time of the functional signature scheme.

1.2 Summary of our Results on Functional Pseudorandom Functions and Selective Pseudorandom Functions

We present formal definitions and constructions of functional pseudorandom functions (F-PRF) and pseudorandom functions with selective access (PRF-SA). In particular, we present a construction based on the existence of one-way functions of a functional pseudorandom function family supporting the class of *prefix-fixing functions*. Our construction is based on the Goldreich-Goldwasser-Micali (GGM) tree-based PRF construction [GGM86].

Theorem 1.6 (Informal). *Assuming the existence of OWF, there exists a functional PRF that supports keys for the following class of functions related to prefix matching: $\mathcal{F}_{\text{pre}} = \{f_z | z \in \{0,1\}^m, m \leq n\}$, where $f_z(x) = x$ if z is a prefix of x , and \perp otherwise. The pseudorandomness property holds against a selective adversary, who declares the functions he will query before seeing the public parameters.*

We remark that one can directly obtain a *fully* secure F-PRF for \mathcal{F}_{pre} , in which security holds against an adversary who adaptively requests key queries, from our selectively secure construction, with a loss of 2^{-n} in security for each functional secret key sk_{f_z} queried by the adversary. This is achieved simply by guessing the adversary's query $f_z \in \mathcal{F}_{\text{pre}}$. For appropriate choices of the input length n , security of the underlying OWF, and number of key queries, this still provides the required security.

Overview of the construction. We show that the original Goldreich-Goldwasser-Micali (GGM) tree-based construction [GGM86] provides the desired functionality, where the functional key sk_f corresponding to a prefix-fixing function $f_z(x) = z_1 z_2 \cdots z_i x_{i+1} \cdots x_n$ will be given by the partial evaluation of the PRF down the tree, at the node corresponding to prefix $z_1 z_2 \cdots z_i$.

This partial evaluation clearly enables a user to compute all possible continuations in the evaluation tree, corresponding to the output of the PRF on any input possessing prefix z . Intuitively, security holds since the other partial evaluations at this level i in the tree still appear random given the evaluation sk_f (indeed, this corresponds to a truncated i -bit input GGM construction).

Punctured pseudorandom functions. Punctured pseudorandom functions [SW13] are a special case of functional PRFs where one can generate keys for the function family $\mathcal{F} = \{f_x(y) = y \text{ if } y \neq x, \text{ and } \perp \text{ otherwise}\}$. Namely, a key for function f_x allows one to compute the pseudorandom function on any input except for x . Punctured PRFs have recently proven useful as one of the main techniques used in proving the security of various cryptographic primitives based on the existence of indistinguishability obfuscation. Some examples include a construction of public-key encryption from symmetric-key encryption and the construction of deniable encryption given by Sahai and Waters in [SW13], as well as an instantiation of random oracles with a concrete hash function for full-domain hash applications by Hohenberger et al. in [HSW13].

We note that the existence of a functional PRF for the prefix-fixing function family gives a construction of punctured PRFs. A key that allows one to compute the PRF on all inputs except $x = x_1 \dots x_n$ consists of n functional keys for the prefix-fixing function family for prefixes: $\bar{x}_1, x_1 \bar{x}_2, x_1 x_2 \bar{x}_3 \dots x_1 x_2 \dots x_{n-1} \bar{x}_n$. We remark that while n prefix-matching keys are revealed, there are only 2^n such *sets* of keys (corresponding to the 2^n choices for the punctured input x), and thus we lose only 2^{-n} security when complexity leveraging from selective to full security. For appropriate choice of underlying OWF security, this yields fully secure punctured PRFs for any desired poly-sized inputs, based on OWFs.

Corollary 1.7. *Assuming the existence of OWF, there exists a (fully) secure punctured PRF for any desired poly-size input length.*

Other notions. Our construction has the additional beneficial property of *hierarchical key generation*: i.e., a party with a functional key sk_{f_z} for a prefix z may generate valid “subordinate” functional keys $\text{sk}_{f_{z'}}$ for any prefix $z' = z|*$. That is, we prove the following additional statement.

Corollary 1.8 (Informal). *Assuming the existence of OWF, there exists a hierarchical functional PRF for the class of functions F_{pre} .*

Recall that we can also view the prefix-matching function as a predicate allowing only signatures of message that begin with a prefix z . As an immediate corollary of the above, we achieve (hierarchical) functional PRFs with *selective access* for the corresponding class of prefix-matching predicates:

Corollary 1.9 (Informal). *Assuming the existence of OWF, there exists a (hierarchical) functional PRF with selective access for the class of prefix-matching predicates $\mathcal{P}_{\text{pre}} = \{P_z | z \in \{0, 1\}^m, m \leq n\}$, where $P_z(x) = 1$ if z is a prefix of x , and 0 otherwise. The pseudorandomness property holds against a selective adversary (or against an adaptive adversary, with a security loss of 2^{-n} per key query).*

1.3 Open Problems

The size of the signatures in our SNARK-based functional signature scheme is dependent only of the security parameter, but it is based on non-falsifiable assumptions. In Section 4, we show that, for a functional signature scheme that supports signing keys for a function f , a signature of $y = f(x)$ cannot be sublinear in the size of y or x , unless the construction is either proven secure under a non-falsifiable assumption or makes use of non black-box techniques. No lower bound exists that relates the size of the signature to the description of f . Constructing functional signatures with short (sublinear in the size of the functions supported) signatures and verification time under falsifiable assumptions remains an open problem.

An interesting problem left open by this work is to construct a functional PRF that is also *verifiable*. A verifiable PRF, introduced by Micali, Rabin and Vadhan in [MRV99] has the property that, in addition to the secret seed of the PRF, there is a corresponding public key and a way to generate a proof π_x given the secret seed, such that given the public key, x , y and π_x one can check that y is indeed the output of the PRF on x . The public parameters and the proof should not allow an adversary to distinguish the outputs of the PRF from random on any point for which the adversary has not received a proof. A construction of standard verifiable PRFs was given by Lysyanskaya based on the many-DH assumption in bilinear groups in [Lys02].

One may extend the notion of verifiable PRFs to the setting of functional PRFs by enabling a user with functional key sk_f to also generate verifiable proofs π_x of correctness for evaluations of the PRF on inputs x for which his key allows. We note that such a verifiable functional pseudorandom function family supporting keys for a function class \mathcal{F} , implies a functional signature scheme that supports signing keys for the same function class, so the lower bound mentioned for functional signatures applies also to the proofs output in the verifiable functional PRF context.

1.4 Other Related Work

Functional Encryption. This work is inspired by recent results on the problem of functional encryption, which was introduced by Sahai and Waters in [SW05], and formalized by Boneh et al. in [BSW11]. In the past few years there has been significant progress on constructing functional encryption schemes for general classes of functions (e.g., [GVW12, GKP⁺12, GKP⁺13]). In this setting, a party with access to a master secret key can generate secret keys for any function f , which allows a third party who has this secret key and an encryption of a message m to learn $f(m)$, but nothing else about m . In [GKP⁺12], Goldwasser et al. construct a functional encryption scheme that can support general functions, where the ciphertext size grows with the maximum depth of the functions for which keys are given. They improve this result in a follow-up work [GKP⁺13], which constructs a functional encryption scheme that supports decryption keys for any Turing machine. Both constructions are secure according to a simulation-based definition, as long as a single key is given out. In [AGVW13], Agrawal et al. show that constructing functional encryption schemes

achieving this notion of security in the presence of an unbounded number of secret keys is impossible for general functions. In contrast, no such impossibility results are known in the setting of functional signatures.

Connections to Obfuscation. The goal of program obfuscation is to construct a compiler O that takes as input a program P and outputs a program $O(P)$ that preserves the functionality of P , but hides all other information about the original program. In [BGI⁺01] Barak et al. formalize this, requiring that, for every adversary having access to an obfuscation of P that outputs a single bit, there exists a simulator that only has blackbox access to P and whose output is statistically close to the adversary’s output:

$$\Pr[A(O(P)) = 1] - \Pr[S^P(1^{|P|}) = 1] = \text{negl}(|P|)$$

Barak et al. [BGI⁺01] construct a class of programs and an adversary for which no simulator can exist, therefore showing that this definition is not achievable for general functions. Furthermore, in [GK05], Goldwasser and Kalai give evidence that several natural cryptographic algorithms, including the signing algorithm of any unforgeable signature scheme, are not obfuscatable with respect to this strong definition.

Consider the function $\text{Sign} \circ f$, where Sign is the signing algorithm of an unforgeable signature scheme, f is an arbitrary function and \circ denotes function composition. Based on the results in [GK05] we would expect this function not to be obfuscatable according to the blackbox simulation definition. A meaningful relaxation of the definition is that, while having access to an obfuscation of this function might not hide all information about the signing algorithm, it does not completely reveal the secret key, and does not allow one to sign messages that are not in the range of f . In our function signature scheme, the signing key corresponding to a function f achieves exactly this definition of security, and we can think of it as an obfuscation of $\text{Sign} \circ f$ according to this relaxed definition. Indeed it has recently come to our attention that Barak in an unpublished manuscript has considered *delegatable signatures*, a highly related concept.

Homomorphic Signatures. Another related problem is that of homomorphic signatures. In a homomorphic signature scheme, a user signs several messages with his secret key. A third party can then perform arbitrary computations over the signed data, and obtain a new signature that authenticates the resulting message with respect to this computation. In [GW12], Gennaro and Wichs construct homomorphic message authenticators, which satisfy a weaker unforgeability notion than homomorphic signatures, in that the verification is done with respect to a secret key unknown to the adversary. They impose an additional restriction on the adversary, who is not allowed to make verification queries. For homomorphic signature schemes with public verification, the most general construction of Boneh and Freeman [BF11] only allows the evaluation of multivariate polynomials on signed data. Constructing homomorphic signature schemes for general functions remains an open problem.

Signatures of correct computation. Papamanthou, Shi and Tamassia considered a notion of functional signatures under the name “signatures of correct computation” in [PST13]. They give constructions for schemes that support operations over multivariate polynomials, such as polynomial evaluation and differentiation. Their constructions are secure in the random oracle model and allow efficient updates to the signing keys: the keys can be updated in time proportional to the number of updated coefficients. In contrast, our constructions that support signing keys for general functions, assuming the existence of succinct non-interactive arguments of knowledge.

Independent work. Finally, as mentioned earlier, related notions to functional PRFs appear in the concurrent and independent works [BW13, KPTZ13]. Based on the Multilinear Decisional Diffie-Hellman assumption (a recently coined assumption related to existence of secure multilinear maps), [BW13] show that PRFs with Selective Access can be constructed for all predicates describable as polynomial-sized circuits. We remark that this is not equivalent to functional PRFs for polynomial-sized circuits, which additionally captures NP relations (i.e., the predicate $y \in \text{Range}(f)$ may not be efficiently testable directly).

Subsequent to our posting of an earlier manuscript of this work, [BF13] and [BMS13] have additionally posted similar results on functional signatures.

1.5 Overview of the paper

In Section 2, we describe several primitives which will be used in our constructions. In Section 3, we give a formal definition of functional signature schemes, and present three constructions satisfying the definition. In Section 4, we show how to construct delegation schemes and succinct non-interactive arguments (SNARGs) from functional signatures schemes. In Section 5, we give a formal definition of functional pseudorandom functions and pseudorandom functions with selective access, and present a sample construction for the prefix-fixing function family.

2 Preliminaries

In this section we define several cryptographic primitives that are used in our constructions.

2.1 Signature Schemes

Definition 2.1. A signature scheme for a message space \mathcal{M} is a tuple $(\text{Gen}, \text{Sign}, \text{Verify})$:

- $\text{Gen}(1^k) \rightarrow (\text{sk}, \text{vk})$: the key generation algorithm is a probabilistic, polynomial-time algorithm which takes as input a security parameter 1^k , and outputs a signing and verification key pair (sk, vk) .
- $\text{Sign}(\text{sk}, m) \rightarrow \sigma$: the signing algorithm is a probabilistic polynomial time algorithm which is given the signing key sk and a message $m \in \mathcal{M}$ and outputs a string σ which we call the signature of m .
- $\text{Verify}(\text{vk}, m, \sigma) \rightarrow \{0, 1\}$: the verification algorithm is a polynomial time algorithm which, given the verification key vk , a message m , and signature σ , returns 1 or 0 indicating whether the signature is valid.

A signature scheme should satisfy the following properties:

Correctness

$$\begin{aligned} \forall (\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^k), \forall m \in \mathcal{M}, \forall \sigma \leftarrow \text{Sign}(\text{sk}, m), \\ \text{Verify}(\text{vk}, m, \sigma) \rightarrow 1 \end{aligned}$$

Unforgeability under chosen message attack

A signature scheme is unforgeable under chosen message attack if the winning probability of any probabilistic polynomial time adversary in the following game is negligible in the security parameter:

- The challenger samples a signing, verification key pair $(\text{sk}, \text{vk}) \leftarrow \text{Gen}(1^k)$ and gives vk to the adversary.
- The adversary requests signatures from the challenger for a polynomial number of messages. In round i , the adversary chooses m_i based on $m_1, \sigma_1, \dots, m_{i-1}, \sigma_{i-1}$, and receives $\sigma_i \leftarrow \text{Sign}(\text{sk}, m_i)$.
- The adversary outputs a signature σ^* and a message m^* , and wins if $\text{Verify}(\text{vk}, m^*, \sigma^*) \rightarrow 1$ and the adversary has not previously received a signature of m^* from the challenger.

Lemma 2.2 ([Rom90]). *Under the assumption that one-way functions exist, there exists a signature scheme which is secure against existential forgery under adaptive chosen message attacks by polynomial-time algorithms.*

2.2 Non-Interactive Zero Knowledge

Definition 2.3. [FLS90, BFM88, BSMP91]: $\Pi = (\text{Gen}, \text{Prove}, \text{Verify}, \mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{Proof}}))$ is an *efficient adaptive NIZK argument system* for a language $L \in \text{NP}$ with witness relation \mathcal{R} if $\text{Gen}, \text{Prove}, \text{Verify}, \mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{Proof}}$ are all PPT algorithms, and there exists a negligible function μ such that for all k the following three requirements hold.

- **Completeness:** For all x, w such that $\mathcal{R}(x, w) = 1$, and for all strings $\text{crs} \leftarrow \text{Gen}(1^k)$,

$$\text{Verify}(\text{crs}, x, \text{Prove}(x, w, \text{crs})) \rightarrow 1.$$

- **Adaptive Soundness:** For all PPT adversaries \mathcal{A} , if $\text{crs} \leftarrow \text{Gen}(1^k)$ is sampled uniformly at random, then the probability that $\mathcal{A}(\text{crs})$ will output a pair (x, π) such that $x \notin L$ and yet $\text{Verify}(\text{crs}, x, \pi) \rightarrow 1$, is at most $\mu(k)$.
- **Adaptive Zero-Knowledge:** For all PPT adversaries \mathcal{A} ,

$$|\Pr[\text{Exp}_{\mathcal{A}}(k) \rightarrow 1] - \Pr[\text{Exp}_{\mathcal{A}}^S(k) \rightarrow 1]| \leq \mu(k),$$

where the experiment $\text{Exp}_{\mathcal{A}}(k)$ is defined by:

$$\begin{aligned} &\text{crs} \leftarrow \text{Gen}(1^k) \\ &\text{Return } \mathcal{A}^{\text{Prove}(\text{crs}, \cdot, \cdot)}(\text{crs}) \end{aligned}$$

and the experiment $\text{Exp}_{\mathcal{A}}^S(k)$ is defined by:

$$\begin{aligned} &(\text{crs}, \text{trap}) \leftarrow \mathcal{S}^{\text{crs}}(1^k) \\ &\text{Return } \mathcal{A}^{S'(\text{crs}, \text{trap}, \cdot, \cdot)}(\text{crs}), \end{aligned}$$

where $S'(\text{crs}, \text{trap}, x, w) = \mathcal{S}^{\text{Proof}}(\text{crs}, \text{trap}, x)$.

We next define the notion of a NIZK argument of knowledge.

Definition 2.4. Let $\Pi = (\text{Gen}, \text{Prove}, \text{Verify}, \mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{Proof}}))$ be an efficient adaptive NIZK argument system for an NP language $L \in \text{NP}$ with a corresponding NP relation \mathcal{R} . We say that Π is a *argument-of-knowledge* if there exists a PPT algorithm $E = (E_1, E_2)$ such that for *every* adversary \mathcal{A} ,

$$|\Pr[\mathcal{A}(\text{crs}) \rightarrow 1 | \text{crs} \leftarrow \text{Gen}(1^k)] - \Pr[\mathcal{A}(\text{crs}) \rightarrow 1 | (\text{crs}, \text{trap}) \leftarrow E_1(1^k)]| = \text{negl}(k)$$

For every PPT adversary \mathcal{A} ,

$$\begin{aligned} &\Pr[\mathcal{A}(\text{crs}) \rightarrow (x, \pi) \text{ and } E(\text{crs}, \text{trap}, x, \pi) \rightarrow w^* \text{ s.t. } \text{Verify}(\text{crs}, x, \pi) \rightarrow 1 \text{ and } (x, w^*) \notin \mathcal{R}] \\ &= \text{negl}(k), \end{aligned}$$

where the probabilities are taken over $(\text{crs}, \text{trap}) \leftarrow E_1(1^k)$, and over the random coin tosses of the extractor algorithm E_2 .

We note that we require the distributions over the honestly generated crs , and the crs generated by the extractor E_1 to be statistically close, whereas they are often required to be just computationally indistinguishable. However, if one is satisfied with computational zero knowledge (as is the case for us), this is actually without loss of generality. Namely, given any NIZKAoK $\Pi = (\text{Gen}, \text{Prove}, \text{Verify}, \mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{Proof}}), E = (E_1, E_2))$ for which the CRS output by $\text{Gen}(1^k)$ and $E_1(1^k)$ are only computationally indistinguishable, we claim that the system Π' formed by using E_1 also as the *honest* CRS generation algorithm (i.e., replacing Gen) is also a NIZKAoK, and satisfies our statistical indistinguishability requirement.

Claim 2.5. Suppose Π as above is a NIZKAoK for which $\{\text{crs} : \text{crs} \leftarrow \text{Gen}(1^k)\} \stackrel{c}{\cong} \{(\text{crs}, \text{trap}) \leftarrow E_1(1^k)\}$ are only computationally indistinguishable. Then $\Pi' := (E_1, \text{Prove}, \text{Verify}, \mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{Proof}}), E = (E_1, E_2))$ is a NIZKAoK as in Definition 2.4.

Proof. Clearly extraction and adaptive soundness are maintained. Completeness must still hold for any statement/witness pairs (x, w) produced by an efficient adversary, due to the computational indistinguishability of CRSs generated by Gen and E_1 ; otherwise there exists an efficient distinguishing algorithm who generates honest proofs and tests whether they verify. Finally, adaptive zero knowledge must hold for the *same* simulator algorithms $\mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{Proof}})$. Indeed, for PPT adversary \mathcal{A} , consider a third experiment $\text{Exp}_{\mathcal{A}}^{\text{E}}(k)$ defined by generating $(\text{crs}, \text{trap}) \leftarrow \text{E}_1(1^k)$, and returning $\mathcal{A}^{\text{Prove}(\text{crs}, \cdot, \cdot)}(\text{crs})$. That is, $\text{Exp}_{\mathcal{A}}^{\text{E}}(k)$ is identical to the real-world experiment $\text{Exp}_{\mathcal{A}}(k)$ except that the CRS is generated according to E_1 instead of Gen (and, in particular, corresponds to the real-world experiment for the modified scheme Π'). By the computational indistinguishability of CRSs generated by Gen and E_1 , it holds that $\text{Exp}_{\mathcal{A}}(k)$ and $\text{Exp}_{\mathcal{A}}^{\text{E}}(k)$ are computationally indistinguishable. But by the adaptive zero knowledge property of the original scheme Π , we have that $\text{Exp}_{\mathcal{A}}(k)$ is computationally indistinguishable from the simulated experiment $\text{Exp}_{\mathcal{A}}^{\text{S}}(k)$. Thus, it must be that $\text{Exp}_{\mathcal{A}}^{\text{E}}(k)$ is computationally indistinguishable from $\text{Exp}_{\mathcal{A}}^{\text{S}}(k)$: that is, Π' satisfies adaptive zero knowledge. \square

Remark. There is a standard way to convert any NIZK argument system Π to a NIZK argument-of-knowledge system Π' . The idea is to append to the crs a public key pk corresponding to any semantic secure encryption scheme. Thus, the common reference string corresponding to Π' is of the form $\text{crs}' = (\text{crs}, \text{pk})$. In order to prove that $x \in L$ using a witness w , choose randomness $r \leftarrow \{0, 1\}^{\text{poly}(k)}$, compute $c \leftarrow \text{Enc}_{\text{pk}}(w, r)$ and compute a NIZK proof π , using the underlying NIZK argument system Π , that $(\text{pk}, x, c) \in L'$, where

$$L' \triangleq \{(\text{pk}, x, c) : \exists(w, r) \text{ s.t. } (x, w) \in \mathcal{R} \text{ and } c \leftarrow \text{Enc}_{\text{pk}}(w, r)\}.$$

Let $\pi' = (\pi, c)$ be the proof.

The common reference string simulator E_1 will generate a simulated crs' by generating $(\text{crs}, \text{trap})$ using the underlying simulator \mathcal{S}^{crs} , and by generating a public key pk along with a corresponding secret key sk . Thus, $\text{trap}' = (\text{trap}, \text{sk})$. The extractor algorithm E_2 , will extract a witness for x from a proof $\pi' = (\pi, c)$ by using sk to decrypt the ciphertext c .

We note that the distribution over the honestly generated crs , and the crs generated by E_1 are statistically close, as required in our definition above.

Lemma 2.6 ([FLS90]). *Assuming the existence of enhanced trapdoor permutations, there exists an efficient adaptive NIZK argument of knowledge for all languages in NP.*

2.3 Succinct Non-Interactive Arguments (SNARGs)

Definition 2.7. $\Pi = (\text{Gen}, \text{Prove}, \text{Verify})$ is a *succinct non-interactive argument* for a language $L \in \text{NP}$ with witness relation \mathcal{R} if it satisfies the following properties:

- **Completeness:** For all x, w such that $\mathcal{R}(x, w) = 1$, and for all strings $\text{crs} \leftarrow \text{Gen}(1^k)$,

$$\text{Verify}(\text{crs}, x, \text{Prove}(x, w, \text{crs})) = 1.$$

- **Adaptive Soundness:** There exists a negligible function $\mu(k)$, such that, for all PPT adversaries \mathcal{A} , if $\text{crs} \leftarrow \text{Gen}(1^k)$ is sampled uniformly at random, then the probability that $\mathcal{A}(\text{crs})$ will output a pair (x, π) such that $x \notin L$ and yet $\text{Verify}(\text{crs}, x, \pi) = 1$, is at most $\mu(k)$.
- **Succinctness:** There exists an universal polynomial $p(\cdot)$ that does not depend on the relation \mathcal{R} , such that

$$\begin{aligned} \forall x, w \text{ s.t. } \mathcal{R}(x, w) = 1, \text{crs} \leftarrow \text{Gen}(1^k), \pi \leftarrow \text{Prove}(x, w, \text{crs}), \\ |\pi| \leq p(k + \log R) \end{aligned}$$

where R denotes the runtime of the relation associated with language L . We note that the definition of succinctness considered in the lower bound of [GW11] is weaker, in that they require the proof size to only be bounded by $r(k) \cdot (|x| + |w|)^{o(1)}$, for some polynomial $r(\cdot)$.

Definition 2.8. A SNARG $\Pi = (\text{Gen}, \text{Prove}, \text{Verify})$ is a *succinct non-interactive argument of knowledge (SNARK)* for a language $L \in \text{NP}$ with witness relation \mathcal{R} if there exists a negligible function $\mu(\cdot)$ such that, for all PPT provers P^* , there exists a PPT algorithm $E_{P^*} = (E_{P^*}^1, E_{P^*}^2)$ such that for *every* adversary \mathcal{A} ,

$$|\Pr[\mathcal{A}(\text{crs}) \rightarrow 1 | \text{crs} \leftarrow \text{Gen}(1^k)] - \Pr[\mathcal{A}(\text{crs}) \rightarrow 1 | (\text{crs}, \text{trap}) \leftarrow E_{P^*}^1(1^k)]| = \mu(k),$$

and,

$$\begin{aligned} &\Pr[P^*(\text{crs}) \rightarrow (x, \pi) \text{ and } E_{P^*}^2(\text{crs}, \text{trap}, x, \pi) \rightarrow w^* \text{ s.t. } \text{Verify}(\text{crs}, x, \pi) \rightarrow 1 \text{ and } (x, w^*) \notin \mathcal{R}] \\ &= \mu(k). \end{aligned}$$

where the probabilities are taken over $(\text{crs}, \text{trap}) \leftarrow E_{P^*}^1(1^k)$, and over the random coin tosses of the extractor algorithm $E_{P^*}^2$.

Remark As in the NIZK definition, we require the distributions over the honestly generated crs , and the crs generated by the extractor $E_{P^*}^1$ to be statistically close. We note that the SNARK construction in [BCCT13] satisfies a stronger definition, where the extraction process has to work for a honestly generated crs , without having access to a trapdoor.

Definition 2.9. A SNARK $\Pi = (\text{Gen}, \text{Prove}, \text{Verify}, E)$ is a *zero-knowledge SNARK* for a language $L \in \text{NP}$ with witness relation \mathcal{R} if there exist PPT algorithms $S = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{Proof}})$ satisfying the following property:

Adaptive Zero-Knowledge: For all PPT adversaries \mathcal{A} ,

$$|\Pr[\text{Exp}_{\mathcal{A}}(k) \rightarrow 1] - \Pr[\text{Exp}_{\mathcal{A}}^S(k) \rightarrow 1]| \leq \mu(k),$$

where the experiment $\text{Exp}_{\mathcal{A}}(k)$ is defined by:

$$\begin{aligned} &\text{crs} \leftarrow \text{Gen}(1^k) \\ &\text{Return } \mathcal{A}^{\text{Prove}(\text{crs}, \cdot, \cdot)}(\text{crs}) \end{aligned}$$

and the experiment $\text{Exp}_{\mathcal{A}}^S(k)$ is defined by:

$$\begin{aligned} &(\text{crs}, \text{trap}) \leftarrow \mathcal{S}^{\text{crs}}(1^k) \\ &\text{Return } \mathcal{A}^{S'(\text{crs}, \text{trap}, \cdot, \cdot)}(\text{crs}), \end{aligned}$$

where $S'(\text{crs}, \text{trap}, x, w) = \mathcal{S}^{\text{Proof}}(\text{crs}, \text{trap}, x)$.

There are several constructions of SNARKs known, all based on non-falsifiable assumptions. A falsifiable assumption is an assumption that can be modeled as a game between an efficient challenger and an adversary. Most standard cryptographic assumptions are falsifiable. This includes both general assumptions like the existence of OWFs, trapdoor predicates, and specific assumptions (discrete logarithm, RSA, LWE, hardness of factoring).

Lemma 2.10 ([BCCT13]). *A SNARK system for any language $L \in \text{NP}$ can be constructed assuming the existence of collision-resistant hash function and knowledge of exponent assumptions.*

Lemma 2.11 ([BCCT12]). *If there exist SNARKs and NIZKAoK for NP, then there exist zero-knowledge SNARKs for all languages in NP.*

In [GW11] Gentry and Wichs show that no construction of SNARKs, with proof size bounded by $r(k) \cdot (|x| + |w|)^{o(1)}$, for some polynomial $r(\cdot)$, can be proved secure under a black-box reduction to a falsifiable assumption. A black-box reduction is one that only uses oracle access to an attacker, and does not use that adversary's code in any other way. The definition of succinctness in [GW11] is a relaxation of the one in definition Definition 2.8, which makes their lower bound result stronger.

2.4 Delegation Schemes

A delegation scheme allows a client to outsource the evaluation of a function F to a server, while allowing the client to verify the correctness of the computation. The verification process should be more efficient than computing the function. We formalize these requirements below, following the definition introduced by Gennaro et al. in [GGP10].

Definition 2.12 ([GGP10]). A *delegation scheme* for a function F consists of a tuple of algorithms (KeyGen, Encode, Compute, Verify)

- **KeyGen**($1^k, F$) \rightarrow (enc, evk, vk): The key generation algorithm takes as input a security parameter k and a function F , and outputs a key **enc** that is used to encode the input, an evaluation key **evk** that is used for the evaluation of the function F , and a verification key **vk** that is used to verify that the output was computed correctly.
- **Encode**(enc, x) $\rightarrow \sigma_x$: The encoding algorithm uses the encoding key **enc** to encode the function input x as a public value σ_x , which is given to the server to compute with.
- **Compute**(evk, σ_x) $\rightarrow (y, \pi_y)$: Using the public evaluation key, **evk** and the encoded input σ_x , the server computes the function output $y = F(x)$, and a proof π_y that y is the correct output.
- **Verify**(vk, x, y, π_y) $\rightarrow \{0, 1\}$: The verification algorithm checks the proof π_y and outputs 1 (indicating that the proof is correct), or 0 otherwise.

We require a delegation scheme to satisfy the following requirements:

Correctness

For all vk, x, y, π_y such that $(\text{enc}, \text{evk}, \text{vk}) \leftarrow \text{KeyGen}(1^k, F)$, $\sigma_x \leftarrow \text{Encode}(\text{enc}, x)$, $(y, \pi_y) \leftarrow \text{Compute}(\text{evk}, \sigma_x)$,

$$\text{Verify}(\text{vk}, x, y, \pi_y) \rightarrow 1$$

Authentication

For all PPT adversaries, the probability that the adversary is successful in the following game is negligible:

- The challenger runs **KeyGen**($1^k, F$) \rightarrow (enc, evk, vk), and gives (evk, vk) to the adversary.
- The adversary gets access to an encoding oracle, $O_{\text{enc}}(\cdot) = \text{Encode}(\text{enc}, \cdot)$.
- The adversary is successful if it can produce a tuple (x, y, π_y) such that $y \neq F(x)$ and $\text{Verify}(\text{vk}, x, y, \pi_y) \rightarrow 1$.

Efficient verification

Let $T(n)$ be the running time of the verification algorithm on inputs of size n . Let $T_F(n)$ be the running time of F on inputs of size n . We require the worst-case running time of the verification algorithm to be sub linear in the worst case running time of F ,

$$T(n) \in o(T_F(n))$$

2.5 Pseudorandom Generators and Functions

Definition 2.13. A *pseudorandom generator* (PRG) is a length expanding function $\text{prg} : \{0, 1\}^k \rightarrow \{0, 1\}^n$ (for $n > k$) such that $\text{prg}(U_k)$ and U_n are computationally indistinguishable, where U_k is a uniformly distributed k -bit string and U_n is a uniformly distributed n -bit string.

Definition 2.14. [GGM86] A family of functions $\mathcal{F} = \{F_s\}_{s \in S}$, indexed by a set S , and where $F_s : D \rightarrow R$ for all s , is a *pseudorandom function (PRF) family* if for a randomly chosen s , and all PPT \mathcal{A} , the distinguishing advantage $\Pr_{s \leftarrow S}[\mathcal{A}^{F_s(\cdot)} = 1] - \Pr_{f \leftarrow (D \rightarrow R)}[\mathcal{A}^{f(\cdot)} = 1]$ is negligible, where $(D \rightarrow R)$ denotes the set of all functions from D to R .

3 Functional Signatures: Definition and Constructions

3.1 Formal Definition

We now give a formal definition of a functional signature scheme, and explain in more detail the unforgeability and function privacy properties a functional signature scheme should satisfy.

Definition 3.1. A *functional signature scheme* for a message space \mathcal{M} , and function family $\mathcal{F} = \{f : \mathcal{D}_f \rightarrow \mathcal{M}\}$ consists of algorithms (FS.Setup, FS.KeyGen, FS.Sign, FS.Verify):

- FS.Setup(1^k) \rightarrow (msk, mvk): the setup algorithm takes as input the security parameter and outputs the master signing key and master verification key.
- FS.KeyGen(msk, f) \rightarrow sk_f : the key generation algorithm takes as input the master signing key and a function $f \in \mathcal{F}$ (represented as a circuit), and outputs a signing key for f .
- FS.Sign(f, sk_f, m) \rightarrow ($f(m), \sigma$): the signing algorithm takes as input the signing key for a function $f \in \mathcal{F}$ and an input $m \in \mathcal{D}_f$, and outputs $f(m)$ and a signature of $f(m)$.
- FS.Verify(mvk, m^*, σ) \rightarrow $\{0, 1\}$: the verification algorithm takes as input the master verification key mvk, a message m and a signature σ , and outputs 1 if the signature is valid.

We require the following conditions to hold:

Correctness:

$$\forall f \in \mathcal{F}, \forall m \in \mathcal{D}_f, (\text{msk}, \text{mvk}) \leftarrow \text{FS.Setup}(1^k), sk_f \leftarrow \text{FS.KeyGen}(\text{msk}, f), (m^*, \sigma) \leftarrow \text{FS.Sign}(f, sk_f, m), \\ \text{FS.Verify}(\text{mvk}, m^*, \sigma) = 1.$$

Unforgeability:

The scheme is unforgeable if the advantage of any PPT algorithm A in the following game is negligible:

- The challenger generates $(\text{msk}, \text{mvk}) \leftarrow \text{FS.Setup}(1^k)$, and gives mvk to A
- The adversary is allowed to query a key generation oracle \mathcal{O}_{key} , and a signing oracle $\mathcal{O}_{\text{sign}}$, that share a dictionary indexed by tuples $(f, i) \in \mathcal{F} \times \mathbb{N}$, whose entries are signing keys: $sk_f^i \leftarrow \text{FS.KeyGen}(\text{msk}, f)$. This dictionary keeps track of the keys that have been previously generated during the unforgeability game. The oracles are defined as follows :
 - $\mathcal{O}_{\text{key}}(f, i)$:
 - * if there exists an entry for the key (f, i) in the dictionary, then output the corresponding value, sk_f^i .
 - * otherwise, sample a fresh key $sk_f^i \leftarrow \text{FS.KeyGen}(\text{msk}, f)$, add an entry $(f, i) \rightarrow sk_f^i$ to the dictionary, and output sk_f^i
 - $\mathcal{O}_{\text{sign}}(f, i, m)$:
 - * if there exists an entry for the key (f, i) in the dictionary, then generate a signature on $f(m)$ using this key: $\sigma \leftarrow \text{FS.Sign}(f, sk_f^i, m)$.
 - * otherwise, sample a fresh key $sk_f^i \leftarrow \text{FS.KeyGen}(\text{msk}, f)$, add an entry $(f, i) \rightarrow sk_f^i$ to the dictionary, and generate a signature on $f(m)$ using this key: $\sigma \leftarrow \text{FS.Sign}(f, sk_f^i, m)$.
- The adversary wins if it can produce (m^*, σ) such that
 - FS.Verify(mvk, m^*, σ) = 1.
 - there does not exist m such that $m^* = f(m)$ for any f which was sent as a query to the \mathcal{O}_{key} oracle.
 - there does not exist a (f, m) pair such that (f, m) was a query to the $\mathcal{O}_{\text{sign}}$ oracle and $m^* = f(m)$.

Function privacy:

Intuitively, we require the distribution of signatures on a message m' generated via different keys sk_f to be computationally indistinguishable, *even given the secret keys and master signing key*. Namely, the advantage of any PPT adversary in the following game is negligible:

- The challenger honestly generates a key pair $(\text{mvk}, \text{msk}) \leftarrow \text{FS.Setup}(1^k)$ and gives both values to the adversary. (Note wlog this includes the randomness used in generation).
- The adversary chooses a function f_0 and receives an (honestly generated) secret key $\text{sk}_{f_0} \leftarrow \text{FS.KeyGen}(\text{msk}, f_0)$.
- The adversary chooses a second function f_1 for which $|f_0| = |f_1|$ (where padding can be used if there is a known upper bound) and receives an (honestly generated) secret key $\text{sk}_{f_1} \leftarrow \text{FS.KeyGen}(\text{msk}, f_1)$.
- The adversary chooses a pair of values m_0, m_1 for which $|m_0| = |m_1|$ and $f_0(m_0) = f_1(m_1)$.
- The challenger selects a random bit $b \leftarrow \{0, 1\}$ and generates a signature on the image message $m' = f_0(m_0) = f_1(m_1)$ using secret key sk_{f_b} , and gives the resulting signature $\sigma \leftarrow \text{FS.Sign}(\text{sk}_{f_b}, m_b)$ to the adversary.
- The adversary outputs a bit b' , and wins the game if $b' = b$.

Succinctness:

There exists a polynomial $s(\cdot)$ such that for every $k \in \mathbb{N}, f \in \mathcal{F}, m \in \mathcal{D}_f$, it holds with probability 1 over $(\text{msk}, \text{mvk}) \leftarrow \text{FS.Setup}(1^k); \text{sk}_f \leftarrow \text{FS.KeyGen}(\text{msk}, f); (f(m), \sigma) \leftarrow \text{FS.Sign}(f, \text{sk}_f, m)$ that the resulting signature on $f(m)$ has size $|\sigma| \leq s(k, |f(m)|)$. In particular, the signature size is independent of the size $|m|$ of the input to the function, and of the size $|f|$ of a description of the function f .

3.2 Construction

In this section, we present a construction of a (succinct) functional signature scheme, based on succinct non-interactive arguments of knowledge (SNARKs).

Theorem 3.2. *Assuming the existence of SNARKs for NP, there exists a succinct, function-private functional signature scheme for the class of polynomial-size circuits.*

We achieve this via two steps. We first give a construction of a weaker functional signature scheme, achieving correctness and unforgeability but not succinctness or function privacy, based on one-way functions. We then show how to use any weak functional signature scheme (satisfying correctness and unforgeability), together with a SNARK system, to obtain a functional signature scheme that is additionally succinct and function-private. In a third construction, we demonstrate that if one does not require the signatures to be succinct (but still demand function privacy), this transformation can be achieved based on non-interactive zero-knowledge arguments of knowledge (NIZKAoKs).

We present these three constructions in the following three subsections.

3.2.1 OWF-based construction

In this section we give a construction of a functional signature scheme from any standard signature scheme (i.e., existentially unforgeable under chosen-message attack). Our constructed functional signature scheme satisfies the unforgeability property given in Definition 3.1, but not function privacy or succinctness. Since standard signature schemes can be based on one-way functions (OWF) [Rom90], this shows that we can also construct functional signature schemes under the assumption that OWFs exist.

The main ideas of the construction are as follows. The master signing and verification keys (msk, mvk) will simply be a standard key pair for the underlying signature scheme. As part of the signing key for a function f , the signer receives a fresh key pair (sk, vk) for the underlying signature scheme, together with a signature (with respect to mvk) on the function f together with vk . We can think of this signature as a certificate authenticating that the owner of key vk has received permission to sign messages in the range of f . We describe the construction below.

Let $\text{Sig} = (\text{Sig.Setup}, \text{Sig.Sign}, \text{Sig.Verify})$ be a signature scheme that is existentially unforgeable under chosen message attack. We construct a functional signature scheme $\text{FS1} = (\text{FS1.Setup}, \text{FS1.KeyGen}, \text{FS1.Sign}, \text{FS1.Verify})$ as follows:

- **FS1.Setup(1^k):**
 - Sample a signing and verification key pair for the standard signature scheme $(\text{msk}, \text{mvk}) \leftarrow \text{Sig.Setup}(1^k)$, and set the master signing key to be msk , and the master verification key to be mvk .
- **FS1.KeyGen(msk, f):**
 - choose a new signing and verification key pair for the underlying signature scheme: $(\text{sk}, \text{vk}) \leftarrow \text{Sig.Setup}(1^k)$.
 - compute $\sigma_{vk} \leftarrow \text{Sig.Sign}(\text{msk}, f|\text{vk})$, a signature of f concatenated with the new verification key vk .
 - create the certificate $c = (f, \text{vk}, \sigma_{vk})$.
 - output $\text{sk}_f = (\text{sk}, c)$.
- **FS1.Sign(f, sk_f, m):**
 - parse sk_f as (sk, c) , where sk is a signing key for the underlying signature scheme, and c is a certificate as described in the **KeyGen** algorithm.
 - sign m using sk : $\sigma_m \leftarrow \text{Sig.Sign}(\text{sk}, m)$.
 - output $(f(m), \sigma)$, where $\sigma = (m, c, \sigma_m)$.
- **FS1.Verify(mvk, m^*, σ):**
 - parse $\sigma = (m, c = (f, \text{vk}, \sigma_{vk}), \sigma_m)$ and check that:
 1. $m^* = f(m)$.
 2. $\text{Sig.Verify}(\text{vk}, m, \sigma_m) \rightarrow 1$: σ_m is a valid signature of m under the verification key vk .
 3. $\text{Sig.Verify}(\text{mvk}, \text{vk}|f, \sigma_{vk}) = 1$: σ_{vk} is a valid signature of $f|\text{vk}$ under the verification key mvk .

Theorem 3.3. *If the signature scheme **Sig** is existentially unforgeable under chosen message attack, **FS1** as specified above satisfies the unforgeability requirement for functional signatures.*

Proof. Fix a PPT adversary \mathcal{A}_{FS} , and let $Q(k)$ be a polynomial upper bound on the number of queries made by \mathcal{A}_{FS} to the oracles \mathcal{O}_{key} and $\mathcal{O}_{\text{sign}}$. We will use \mathcal{A}_{FS} to construct an adversary \mathcal{A}_{sig} such that, if \mathcal{A}_{FS} wins in the unforgeability game for functional signatures with non-negligible probability, then \mathcal{A}_{sig} breaks the underlying signature scheme, which is assumed to be secure against chosen message attack.

For \mathcal{A}_{FS} to win the functional signature unforgeability game, it must produce a message signature pair (m^*, σ) , where $\sigma = (m, (f, \text{vk}, \sigma_{vk}), \sigma_m)$ such that:

- σ_m is a valid signature of m under the verification key vk .
- σ_{vk} is a valid signature of $f|\text{vk}$ under mvk .
- $f(m) = m^*$.
- \mathcal{A}_{FS} has not sent a query of the form $\mathcal{O}_{\text{key}}(\tilde{f}, i)$ to the signing key generation oracle for any \tilde{f} that has m^* in its range.
- \mathcal{A}_{FS} hasn't sent a query of the form $\mathcal{O}_{\text{sign}}(\tilde{f}, i, \tilde{m})$ to the signing oracle for any \tilde{f}, \tilde{m} such that $\tilde{f}(\tilde{m}) = m^*$.

There are two cases for such a forgery (m^*, σ) , where $\sigma = (m, (f, \text{vk}, \sigma_{vk}), \sigma_m)$:

- **Type I forgery:** The values (f, vk) are such that the concatenated pair $f|\text{vk}$ has *not* already been signed under mvk during any point of the signing and key oracle queries during the security game.
- **Type II forgery:** The values (f, vk) are such that the concatenated pair $f|\text{vk}$ *has* been signed under mvk during the course of \mathcal{A}_{FS} 's oracle queries.

Here we refer to all mvk signatures generated by the oracles $\mathcal{O}_{\text{sign}}$, \mathcal{O}_{key} as intermediate steps in order to answer \mathcal{A}_{FS} 's respective queries.

We now describe the constructed signature adversary, \mathcal{A}_{sig} . In the security game for the standard (existentially unforgeable under chosen message attack) signature scheme, \mathcal{A}_{sig} is given the verification key vk_{sig} ,

and access to a signing oracle $\mathcal{O}_{\text{RegSig}}$. He is considered to be successful in producing a forgery if he outputs a valid signature for a message that was not queried from $\mathcal{O}_{\text{RegSig}}$.

\mathcal{A}_{sig} interacts with \mathcal{A}_{FS} , playing the role of the challenger in the security game for the functional signature scheme. This means that \mathcal{A}_{sig} must simulate the \mathcal{O}_{key} and $\mathcal{O}_{\text{sign}}$ oracles. \mathcal{A}_{FS} flips a coin b , indicating his guess for the type of forgery \mathcal{A}_{FS} will produce, and places his challenge accordingly.

Case 1: $b = 1$: \mathcal{A}_{sig} guesses that \mathcal{A}_{FS} will produce a **Type I** forgery:

First \mathcal{A}_{sig} forwards his challenge verification key vk_{sig} to \mathcal{A}_{FS} as the master verification key in the functional signature security game.

To simulate the \mathcal{O}_{key} , and $\mathcal{O}_{\text{sign}}$ oracles, \mathcal{A}_{sig} maintains a dictionary indexed by tuples (f, i) , whose entries are signing keys for the functional signature scheme that have already been generated. \mathcal{A}_{sig} answers the queries issued by \mathcal{A}_{FS} as follows:

- $\mathcal{O}_{\text{key}}(f, i)$:
 - if there exists an entry for the key (f, i) in the dictionary, then output the corresponding value, sk_f^i .
 - otherwise, \mathcal{A}_{sig} generates a new key pair for the underlying signature scheme, $(\text{sk}, \text{vk}) \leftarrow \text{Sig.Setup}(1^k)$, obtains $\sigma_{\text{vk}} \leftarrow \mathcal{O}_{\text{RegSig}}(f|\text{vk})$ from its own signing oracle (in the standard signature challenge), and returns $\text{sk}_f = (\text{sk}, \sigma_{\text{vk}})$ to \mathcal{A}_{FS} . It also sets entry (f, i) in its dictionary to sk_f .
- $\mathcal{O}_{\text{sign}}(f, i, m)$:
 - if there exists an entry for the key (f, i) in the dictionary, $\text{sk}_f^i = (\text{sk}, \sigma_{\text{vk}})$. It then generates a signature using sk_f^i : that is, generate a signature $\sigma_m \leftarrow \text{Sig.Sign}(\text{sk}, m)$, and output $(f(m), \sigma)$, where $\sigma = (m, c = (f, \text{vk}, \sigma_{\text{vk}}), \sigma_m)$.
 - otherwise, \mathcal{A}_{sig} generates a new key pair for the regular signature scheme, $(\text{sk}, \text{vk}) \leftarrow \text{Sig.Setup}(1^k)$, obtains $\sigma_{\text{vk}} \leftarrow \mathcal{O}_{\text{RegSig}}(f|\text{vk})$ from its signing oracle, and sets entry (f, i) in its dictionary to $\text{sk}_f = (\text{sk}, \sigma_{\text{vk}})$. It then generates $\sigma_m \leftarrow \text{Sig.Sign}(\text{sk}, m)$, and outputs $(f(m), \sigma)$, where $\sigma = (m, c = (f, \text{vk}, \sigma_m), \sigma_{\text{vk}})$.

Eventually, \mathcal{A}_{FS} outputs a signature (m^*, σ) , where $\sigma = (m, (f, \text{vk}, \sigma_{\text{vk}}), \sigma_m)$. \mathcal{A}_{sig} outputs $(f|\text{vk}, \sigma_{\text{vk}})$ as its message-forgery pair in the security game for the standard signature scheme.

Case 2: $b = 0$: \mathcal{A}_{sig} guesses that \mathcal{A}_{FS} will produce a **Type II** forgery:

\mathcal{A}_{sig} generates a new key pair $(\text{msk}, \text{mvk}) \leftarrow \text{Sig.Setup}(1^k)$ himself, and forwards mvk to \mathcal{A}_{FS} . He also guesses a random index q between 1 and $Q(k)$, denoting which of \mathcal{A}_{FS} 's signing queries he will embed his challenge verification key in. He keeps track of the number of keys generated so far in a variable NUMKEYS , which is initialized to 0. As before, \mathcal{A}_{sig} maintains a dictionary indexed by tuples (f, i) , whose entries are signing keys for the functional signature scheme that have already been generated. \mathcal{A}_{sig} answers the queries issued by \mathcal{A}_{FS} as follows:

- $\mathcal{O}_{\text{key}}(f, i)$:
 - if there exists an entry for the key (f, i) in the dictionary, with value *CHALLENGE*, abort
 - if there exists an entry for the key (f, i) in the dictionary and its value is not *CHALLENGE*, then output the corresponding value, sk_f^i .
 - otherwise, \mathcal{A}_{sig} generates a new key pair for the regular signature scheme, $(\text{sk}, \text{vk}) \leftarrow \text{Sig.Setup}(1^k)$, generates $\sigma_{\text{vk}} \leftarrow \text{Sign}(\text{msk}, f|\text{vk})$ himself, and returns $\text{sk}_f = (\text{sk}, \sigma_{\text{vk}})$ to \mathcal{A}_{FS} . It also sets entry (f, i) in its dictionary to sk_f .
- $\mathcal{O}_{\text{sign}}(f, i, m)$:
 - if there exists an entry for the key (f, i) in the dictionary, $\text{sk}_f^i = (\text{sk}, \sigma_{\text{vk}})$, generate $\sigma_m \leftarrow \text{Sig.Sign}(\text{sk}, m)$, and output $(f(m), \sigma)$, where $\sigma = (m, c = (f, \text{vk}, \sigma_{\text{vk}}), \sigma_m)$.
 - if there is no (f, i) entry in the dictionary, and $\text{NUMKEYS} \neq q$, \mathcal{A}_{sig} generates a new key pair for the regular signature scheme, $(\text{sk}, \text{vk}) \leftarrow \text{Sig.Setup}(1^k)$, signs $f|\text{vk}$ himself with respect

to msk : $\sigma_{vk} \leftarrow \text{Sig.Sign}(\text{msk}, f|vk)$, and sets entry (f, i) in its dictionary to sk_f . It then generates a signature on m with respect to the new key sk : $\sigma_m \leftarrow \text{Sig.Sign}(\text{sk}, m)$, and outputs $(f(m), \sigma)$, where $\sigma = (m, c = (f, vk, \sigma_{vk}), \sigma_m)$. NUMKEYS is then incremented.

- if there is no (f, i) entry in the dictionary and $\text{NUMKEYS} = q$, or if the (f, i) entry in the dictionary is set to *CHALLENGE*, then A_{sig} queries its oracle for a signature of m under vk_{sig} , $\sigma_m \leftarrow O_{\text{Reg}_{\text{sig}}}(m)$, computes $\sigma_{vk} \leftarrow \text{Sig.Sign}(\text{msk}, f|vk_{\text{sig}})$, and outputs $(f(m), \sigma)$, where $\sigma = (m, c = (f, vk, \sigma_{vk}), \sigma_m)$. If there is no (f, i) entry in the dictionary, A_{sig} sets it to *CHALLENGE*. NUMKEYS is then incremented.

If A_{sig} does not abort, A_{FS} will eventually output a signature (m^*, σ) , where $\sigma = (m, (f, vk, \sigma_{vk}), \sigma_m)$. A_{sig} outputs (m, σ_m) as its forgery in the security game for the standard signature scheme with respect to vk_{sig} .

We will now argue that if A_{FS} forges in the functional signature scheme with non-negligible probability then A_{sig} is wins the unforgeability game for the standard signature scheme with non-negligible probability.

First note that as long as A_{sig} does not abort (i.e., the bad situation is not encountered where the adversary requests the secret key corresponding to the embedded vk_{sig} challenge), then his answers to the A_{FS} 's keygen and signing queries are simulated perfectly as in the real world. Further, as long as there is not an abort, the view of A_{FS} is independent of A_{sig} 's choice of b and q , as they only determine which verification key is the challenge verification key vk_{sig} .

Now, if A_{FS} produces a **Type I forgery**, then by definition this forgery must include a signature on a new message $f|vk$ that was *not ever signed* under the master verification key mvk during the course of any oracle query response. Thus, if A_{FS} makes a Type I forgery and A_{sig} guessed $b = 1$ (embedding his challenge signature key in the position of the mvk), then A_{FS} 's forgery includes a signature on a new message $f|vk$ that A_{sig} did not query to his signature oracle, constituting a forger in the unforgeability game for the standard signature scheme.

If A_{FS} produces a **Type II forgery**, then the corresponding $f|vk$ was already signed under the master verification key mvk during the course of one of the oracle queries. This cannot have occurred during a O_{key} query, as it would mean that A_{FS} queried O_{key} on the function f , and producing a signature with respect to this f is not a valid forgery in the functional signature scheme. It must then have been signed during an O_{sign} query. Namely, the verification key vk must have been freshly generated during a query of the form $O_{\text{sign}}(f, i, m)$ for which no entry under index (f, i) previously existed, and then the pair $f|vk$ was signed.

Note that if A_{FS} produces a Type II forgery and A_{sig} guessed $b = 0$ and the correct q to embed his challenge, and A_{sig} does not abort, the forgery produced by A_{FS} must include a signature of a new message \tilde{m} with respect to vk_{sig} , for a \tilde{m} that A_{sig} hasn't queried from his signing oracle, and therefore A_{sig} can use this forgery as its own forged signature in the unforgeability game for the standard signature scheme.

We note that, if A_{sig} does abort, it must be that he embedded his challenge in a query q of the form $O_{\text{sign}}(f, i, m)$, and later A_{FS} issued a key generation query $O_{\text{sign}}(f, i)$. But this query can't be the signing query q^* for which the adversary receives a signature of $f|vk$ under mvk , and later outputs a signature of $f(m')$ for another m' . Since the adversary has queried the $O_{\text{sign}}(f, i)$, no message in the range of f would be considered a forgery in the functional signature game. We can conclude that, if A_{sig} aborts, he didn't guess q^* correctly, so we don't need to consider this case separately.

Denoting by b, q the guesses of A_{sig} , we have that success probability of if A_{sig} is therefore:

$$\begin{aligned}
& \Pr[\mathbf{A}_{\text{sig}} \text{ forges in signature challenge}] \\
& \geq \Pr[b = 1 \wedge \mathbf{A}_{\text{FS}} \text{ outputs Type I forgery}] \\
& \quad + \sum_{q^* \in [Q(k)]} \Pr[b = 0 \wedge q = q^* \wedge \mathbf{A}_{\text{sig}} \text{ does not abort} \wedge \mathbf{A}_{\text{FS}} \text{ outputs Type II forgery wrt } \mathbf{vk}_{q^*}] \\
& = \Pr[b = 1 \wedge \mathbf{A}_{\text{FS}} \text{ outputs Type I forgery}] \\
& \quad + \sum_{q^* \in [Q(k)]} \Pr[b = 0 \wedge q = q^* \wedge \mathbf{A}_{\text{FS}} \text{ outputs Type II forgery on } \mathbf{vk}_{q^*}] \\
& \geq \frac{1}{2} \Pr[\mathbf{A}_{\text{FS}} \text{ outputs Type I forgery}] + \frac{1}{2Q(k)} \sum_{i^* \in [Q(k)]} \Pr[\mathbf{A}_{\text{FS}} \text{ outputs Type II forgery on } \mathbf{vk}_{q^*}] \\
& \geq \frac{1}{2Q(k)} \left[\Pr[\mathbf{A}_{\text{FS}} \text{ outputs Type I forgery}] + \sum_{i^* \in [Q(k)]} \Pr[\mathbf{A}_{\text{FS}} \text{ outputs Type II forgery on } \mathbf{vk}_{q^*}] \right] \\
& = \frac{1}{2Q(k)} \Pr[\mathbf{A}_{\text{FS}} \text{ forges}]
\end{aligned}$$

Thus, if \mathbf{A}_{FS} produces a forgery in the functional signature scheme with non-negligible probability $1/P(k)$, then \mathbf{A}_{sig} successfully forges in the underlying signature scheme with non-negligible probability $1/2P(k)Q(k)$. But, this cannot be the case, since we've assumed that \mathbf{Sig} is existentially unforgeable against chosen-message attack. We conclude that $\mathbf{FS1}$ satisfies the unforgeability requirement for functional signatures. \square

While this construction is secure under a very general assumption (the existence of one-way functions), it does not provide function privacy guarantees (indeed, the signature *contains* a description of the relevant pre image and function), and its efficiency can be greatly improved. The size of a signature generated with key \mathbf{sk}_f ($\sigma \leftarrow \mathbf{FS}.\text{Sign}(\mathbf{sk}_f, m)$) in this scheme is proportional to the size of $|f| + |m|$ plus the size of a signature of the standard signature scheme. In contrast, we will next show how to use SNARKs to construct a functional signature where the signature size is proportional to $|f(m)|$, instead of $|f| + |m|$.

3.2.2 Succinct, Function-Private Functional Signatures from SNARKs

We demonstrate how to combine any unforgeable functional signature scheme (such as the OWF-based construction from the previous section) together with a succinct non-interactive argument of knowledge (SNARK) to obtain a new functional signature scheme also satisfying *succinctness and function privacy*.

Let $\mathbf{FS1} = (\mathbf{FS1}.\text{Setup}, \mathbf{FS1}.\text{Sign}, \mathbf{FS1}.\text{Verify})$ be a functional signature scheme, satisfying the unforgeability game as in Definition 3.1, but not necessarily function privacy or succinctness. Let $\Pi = (\text{Gen}, \text{Prove}, \text{Verify})$, $\mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{Proof}})$, $\mathbf{E} = (\mathbf{E}_1, \mathbf{E}_2)$ be an efficient adaptive zero-knowledge SNARK system for the following NP language L :

$$L = \{(m, \mathbf{mvk}) \mid \exists \sigma \text{ s.t. } \mathbf{FS1}.\text{Verify}(\mathbf{mvk}, m, \sigma) = 1\}.$$

We show how to use $\mathbf{FS1}$ and Π to construct a new functional signature scheme that also satisfies function privacy and succinctness.

- $\mathbf{FS2}.\text{Setup}(1^k)$:
 - choose a master signing key, verification key pair for $\mathbf{FS1}$: $(\mathbf{msk}', \mathbf{mvk}') \leftarrow \mathbf{FS1}.\text{Setup}(1^k)$.
 - choose a crs for the zero-knowledge SNARK: $\text{crs} \leftarrow \Pi.\text{Gen}(1^k)$.
 - set the master secret key $\mathbf{msk} = \mathbf{msk}'$, and the master verification key $\mathbf{mvk} = (\mathbf{mvk}', \text{crs})$.
- $\mathbf{FS2}.\text{KeyGen}(\mathbf{msk}, f)$:

- the key generation algorithm is the same as in the underlying functional signature scheme: $\text{sk}_f \leftarrow \text{FS1.KeyGen}(\text{msk}, f)$.
- $\text{FS2.Sign}(f, \text{sk}_f, m)$:
 - generate a signature on m in the underlying functional signature scheme: $\sigma' \leftarrow \text{FS1.Sign}(f, \text{sk}_f, m)$.
 - generate $\pi \leftarrow \Pi.\text{Prove}((f(m), \text{mvk}'), \sigma', \text{crs})$, a zero-knowledge SNARK that $(f(m), \text{mvk}') \in L$, where L is defined as above, and output $(m^* = f(m), \sigma = \pi)$. Informally, π is a proof that the signer knows a signature of $f(m)$ in the underlying functional signature scheme.
- $\text{FS2.Verify}(\text{mvk}, m^*, \sigma)$:
 - output $\Pi.\text{Verify}(\text{crs}, m^*, \sigma)$: i.e., verify that σ is a valid argument of knowledge of a signature of $f(m)$ in the underlying functional signature scheme.

Theorem 3.4. *Assume the existence of an unforgeable (but not necessarily succinct or function-private) functional signature scheme FS1 supporting the class \mathcal{F} of polynomial-sized circuits, and Π be an adaptive zero-knowledge SNARK system for NP. Then there exists succinct, function-private functional signatures for \mathcal{F} .*

Proof of unforgeability

Suppose there exists an adversary A_{FS2} that produces a forgery in the new functional signature scheme with non-negligible probability. We show how to construct an adversary A_{FS1} that uses A_{FS2} to produce a forgery in the underlying functional signature scheme.

A_{FS1} plays the role of the challenger in the security game for A_{FS2} . He gets a verification key mvk_{FS1} in his own unforgeability game, generates $(\text{crs}, \text{trap}) \leftarrow \text{E}_1(1^k)$, a *simulated* CRS for the ZK-SNARK, together with a trapdoor, and forwards $\text{mvk}_{\text{FS2}} = (\text{mvk}_{\text{FS1}}, \text{crs})$ to A_{FS2} as the new master verification key. A_{FS2} makes two types of queries:

- $\text{O}_{\text{keyFS2}}(f, i)$, which A_{FS1} answers (honestly) by forwarding them to its KeyGen oracle, $\text{O}_{\text{keyFS1}}(f, i)$
- $\text{O}_{\text{signFS2}}(f, m, i)$, in which case A_{FS1} forwards the query to his signing oracle, and receives a signature $\sigma_{\text{FS1}} \leftarrow \text{O}_{\text{signFS1}}(f, m, i)$. It then outputs $\pi \leftarrow \Pi.\text{Prove}((f(m), \text{mvk}_{\text{FS1}}), \sigma, \text{crs})$ as his signature of $f(m)$.

After querying the oracles, A_{FS2} will output an alleged forgery in the functional signature scheme, π^* , on some message m^* . A_{FS1} runs the extractor $\text{E}_2(\text{crs}, \text{trap}, (m^*, \text{mvk}_{\text{FS1}}), \pi^*)$ to recover a witness $w = \sigma$ such that $\text{FS1.Verify}(\text{mvk}_{\text{FS1}}, m^*, \sigma) = 1$. A_{FS1} then submits σ as a forgery in his own unforgeability game.

We now prove that if A_{FS2} forges with noticeable probability, then A_{FS1} also forges with noticeable probability in his own security game.

Hybrid 0. The real-world functional signature challenge experiment. Namely, the CRS is generated in the honest fashion $\text{crs} \leftarrow \text{Gen}(1^k)$, and the adversary's signing queries are answered honestly. Denote the probability of the adversary producing a valid forgery in the functional signature FS2 scheme within this experiment by Forge_0 .

Hybrid 1. The same experiment as Hybrid 0, except the CRS is generated using the extraction-enabling procedure, $(\text{crs}, \text{trap}) \leftarrow \text{E}_1(1^k)$. The remainder of the experiment continues as before with respect to crs . Denote the probability of the adversary producing a valid forgery in the functional signature scheme within this experiment by Forge_1 .

Hybrid 2. The interaction with the adversary is the same as in Hybrid 1. Denote by M the set of all messages signed with msk_{FS1} in the underlying functional signature scheme during the course of the experiment, as a result of A_{FS1} 's key and signing oracle queries. At the experiment conclusion, the ZK-SNARK extraction algorithm is executed on the adversary's alleged forgery π^* (on message m^*) in the functional signature scheme: i.e., $(\sigma^*) \leftarrow \text{E}_2(\text{crs}, \text{trap}, (m^*, \text{mvk}_{\text{FS1}}), \pi^*)$.

Denote by Extract_2 the probability that σ^* is a valid signature in the underlying functional signature scheme FS1 on a message m^* such that $f^* \notin M$. Note that this corresponds to the probability of A_{FS1} successfully producing a forgery.

Unforgeability of the functional signature scheme follows from the following sequence of lemmas.

Lemma 3.5. $\text{Forge}_0 \leq \text{Forge}_1 + \text{negl}(k)$.

Proof. Follows directly from the fact that the CRS values generated via the standard algorithm Gen and those generated by the extraction-enabling algorithm E_1 are *statistically* close, as per Definition 2.8.

More formally, suppose there exists a PPT adversary \mathcal{A} for which $\text{Forge}_1 < \text{Forge}_0 - \epsilon$ for some ϵ . Then the following (not necessarily efficient) adversary \mathcal{A}_{crs} distinguishes between CRS values with advantage ϵ . In the CRS challenge, \mathcal{A}_{crs} is given a value crs (generated by either the standard algorithm or the extraction-enabling algorithm). First, \mathcal{A}_{crs} generates a key pair $(\text{msk}_{\text{FS1}}, \text{mvk}_{\text{FS1}}) \leftarrow \text{FS1.Setup}(1^k)$ for the underlying functional signature scheme, and sends $\text{mvk}_{\text{FS2}} = (\text{mvk}_{\text{FS1}}, \text{crs})$ to \mathcal{A} . He answers \mathcal{A}_{FS2} 's queries as in Hybrid 0, generating signatures and proofs as required (note that \mathcal{A} holds the master secret key msk_{FS1} , which allows him to answer the queries). At the conclusion of \mathcal{A}_{FS2} 's query phase, he outputs an alleged forgery π^* in the functional signature scheme. The adversary \mathcal{A}_{crs} tests whether π^* is indeed a forgery. We note that this verification process might not be efficient, since \mathcal{A}_{crs} needs to test whether the message whose signature \mathcal{A}_{FS2} claims to have forged is actually not in the range of any of the functions f that \mathcal{A}_{FS2} has requested signing keys for. If the forgery verifies, \mathcal{A}_{crs} outputs “standard crs”; otherwise, he outputs “extractable crs”. His advantage in the CRS distinguishing game is precisely $\text{Forge}_1 - \text{Forge}_0$, as desired. Since the real and simulated CRS strings are supposed to be statistically close, the distinguishing advantage $\text{Forge}_1 - \text{Forge}_0$ has to be negligible even for an inefficient adversary. \square

Lemma 3.6. $\text{Forge}_1 \leq \text{Extract}_2 + \text{negl}(k)$.

Proof. This holds by the extraction property of the ZK-SNARK system (Definition 2.8).

Namely, if there exists a PPT adversary \mathcal{A} for which $\text{Forge}_1 > \text{Extract}_2 + \epsilon$ for some ϵ , then the following adversary \mathcal{A}_{Ext} successfully produces a properly-verifying proof π for which extraction fails with probability ϵ (which must be negligible by the SNARK extraction property).

\mathcal{A}_{Ext} receives a CRS value crs generated via $(\text{crs}, \text{trap}) \leftarrow \text{E}_1(1^k)$. He samples a key pair $(\text{msk}_{\text{FS1}}, \text{mvk}_{\text{FS1}}) \leftarrow \text{FS1.Setup}(1^k)$ for the underlying functional signature scheme, sends $\text{mvk}_{\text{FS2}} = (\text{mvk}_{\text{FS1}}, \text{crs})$ to the adversary \mathcal{A} , and answers all of \mathcal{A} 's key and signing oracle queries as in Hybrid 1.

Now, let M the collection of all messages f which were signed by \mathcal{A}_{Ext} during the course of the interaction with \mathcal{A} . Suppose that π^* is a valid forgery on m^* in the functional signature scheme; in particular, π^* is a valid proof that $(m^*, \text{mvk}_{\text{FS1}}) \in L$. We argue that if extraction succeeds on π^* (i.e if $\sigma^* \leftarrow \text{E}_2(\text{crs}, \text{trap}, (m^*, \text{vk}), \pi^*)$ yields a valid witness for $(m^*, \text{vk}) \in L$), then it must be that the extracted σ^* is a valid signature on a message $g \notin M$ in the underlying functional signature scheme, so that we are in the event corresponding to Extract_3 . That is, we show $\text{Forge}_1 - \text{Extract}_2$ is bounded above by the probability that extraction *fails*.

Since π^* is a valid forgery in the functional signature scheme FS2, it must be that $m^* \notin \text{Range}(g)$ for all key queries $\text{O}_{\text{key}}(g, i)$ made by \mathcal{A} , and that $m^* \neq g(x)$ for all signing queries $\text{O}_{\text{sign}}(g, x, i)$ made by \mathcal{A} . Now, if the extracted tuple $(f^*, m, \sigma^*) \leftarrow \text{E}_2(\text{crs}, \text{trap}, (m^*, \text{vk}), \pi^*)$ is a valid witness for $(m^*, \text{vk}) \in L$, then from the definition of the language L it means that $m^* = f^*(m)$ and that σ^* is a valid signature on f^* with respect to the master signing key sk (i.e., $\text{Verify}(\text{vk}, \sigma^*, f^*) = 1$). Recall that the set M consists exactly of the functions g for which \mathcal{A} made a key query, and the collection of *constant functions* $g' \equiv g(x)$ for which \mathcal{A} make a signing query (g, x) . But since $m^* \in \text{Range}(f^*)$ and $m^* \notin \text{Range}(g)$ for all $g \in M$, it must be that $f^* \notin M$, as desired.

Therefore, with probability at least $\text{Forge}_2 - \text{Extract}_3 = \epsilon$, it must hold that π^* is a valid proof but that the extraction algorithm *fails* to extract a valid witness from π^* . By the extraction property of the SNARK system, it must be that ϵ is negligible. \square

Lemma 3.7. $\text{Extract}_2 < \text{negl}(k)$.

Proof. This holds by the unforgeability of the underlying functional signature scheme FS1, since Extract_2 is precisely the probability that adversary \mathcal{A}_{FS1} constructed above produces a successful forgery in the unforgeability game for FS1. \square

Proof of function privacy

We show that any adversary A_{priv} who succeeds in the function privacy game with noticeable advantage can be used to break the zero knowledge property of the ZK-SNARK scheme. Recall that in the adaptive zero knowledge security game, the adversary is given a CRS (either honestly generated or simulated) and access to an oracle who accepts statement-witness pairs (x, w) and responds with either honestly generated or simulated proofs of the statement.

More specifically, consider the following two hybrid experiments:

Hybrid 0. The real function privacy challenge. In particular, the CRS for the ZK-SNARK system is generated honestly as $\text{crs} \leftarrow \Pi.\text{Gen}(1^k)$. The challenge signature, on message m_b for randomly chosen $b \leftarrow \{0, 1\}$ (with respect to key \mathbf{t}_b), is generated by first generating a signature on m_b in the underlying functional signature scheme $\sigma \leftarrow \text{Sig}.\text{Sign}(\text{sk}_{f_b}, m_b)$ and then honestly generating a proof $\pi \leftarrow \Pi.\text{Prove}((f_b(m_b), \text{mvk}), \sigma, \text{crs})$.

Hybrid 1. Similar to Hybrid 0, except that the SNARK appearing in the challenge signature is replaced by a *simulated* argument. Namely, the CRS is generated using the simulator algorithm $(\text{crs}, \text{trap}) \leftarrow \mathcal{S}^{\text{crs}}(1^k)$. And the challenge signature is generated by sampling a random bit $b \leftarrow \{0, 1\}$ and *ignoring* it, instead using the simulator $\pi \leftarrow \mathcal{S}^{\text{Proof}}(\text{crs}, \text{trap}, (m', \text{mvk}))$, where $m' = f_0(m_0) = f_1(m_1)$.

Denote by $\text{win}_0, \text{win}_1$ the advantage of the adversary A_{priv} in guessing the bit b in Hybrid 0 and 1, respectively. Function privacy of FS2 follows from the following two claims.

Claim 3.8. $\text{win}_1 \geq \text{win}_0 - \text{negl}(k)$.

Proof. Follows directly from the adaptive zero knowledge property of the ZK-SNARK system. More explicitly, consider the following adversary A_{ZK} :

1. A_{ZK} receives a CRS value crs from the adaptive zero knowledge challenger (either honestly generated or simulated). In addition, he generates a master key pair for the underlying functional signature scheme: $(\text{msk}, \text{mvk}) \leftarrow \text{FS1}.\text{Setup}(1^k)$. A_{ZK} takes $\text{mvk}' = (\text{mvk}, \text{crs})$ and sends the key pair $(\text{mvk}', \text{msk})$ to the function privacy adversary A_{priv} .
2. A_{priv} responds (adaptively) with function queries f_0, f_1 and a message pair m_0, m_1 with $f_0(m_0) = f_1(m_1)$. For each function query f_b , A_{ZK} generates a corresponding key $\text{sk}_{f_b} \leftarrow \text{FS1}.\text{KeyGen}(\text{msk}, f_b)$ and sends sk_{f_b} to A_{priv} .
3. A_{ZK} prepares the function privacy challenge signature as follows. First, he chooses a random bit $b \leftarrow \{0, 1\}$, and uses $(f_b, m_b, \text{sk}_{f_b})$ to generate a signature on $f_b(m_b)$ in the underlying functional signature scheme: $\sigma \leftarrow \text{FS1}.\text{Sign}(\text{sk}_{f_b}, m_b)$. He then submits the query $((f_b(m_b), \text{mvk}), \sigma)$ to the proof oracle in his own ZK challenge. (Recall that σ is a valid witness for $(f_b(m_b), \text{mvk}) \in L$). Denote the oracle response by π , which is either honestly generated or simulated.
4. A_{ZK} sends the signature π to A_{priv} , who responds with a guessed bit b' in the function privacy game. If $b' = b$, then A_{ZK} outputs “real.” Otherwise, if $b' \neq b$, then A_{ZK} outputs “simulated.”

Note that if A_{ZK} has access to the Real Proof experiment (Experiment $\text{Exp}_A(k)$ in Definition 2.9), then A_{ZK} perfectly simulates Hybrid 0, whereas if he has access to the Simulated Proof experiment (Experiment $\text{Exp}_A^S(k)$ in Definition 2.9), then A_{ZK} perfectly simulates Hybrid 1. Thus, A_{ZK} 's advantage in the adaptive zero knowledge challenge is equal to $\text{win}_0 - \text{win}_1$, which by the ZK security of the ZK-SNARK scheme must hence be negligible. □

Claim 3.9. $\text{win}_1 < \text{negl}(k)$.

Proof. Note that the view of A_{priv} in Hybrid 1 is in fact *independent* of the selected bit b . Indeed, the challenge signature is generated with respect *only* to the value $m' = f_0(m_0) = f_1(m_1)$, and not any particular witness. Thus, information theoretically, even a computationally unbounded adversary could not correctly guess the bit b with noticeable advantage. □

Succinctness

The succinctness of our signature scheme follows directly from the succinctness property of the SNARK system. Namely, the size of a functional signature produced by $\text{FS2.Sign}(f, \text{sk}_f, m)$ is exactly the proof length of a SNARK for the language L . There exists a polynomial q such that the runtime R of the associated relation is bounded by $q(|f(m)| + |\text{mvk}| + |\sigma|)$, where σ is a signature in the underlying, non-succinct functional signature scheme.

By Definition 2.7, there exists a polynomial p , such that the corresponding proof length is bounded by $p(k + \text{polylog}(|f(m)| + |\text{mvk}| + |\sigma|))$. The size of the signature $|\sigma| = \text{poly}(|f| + |m| + k)$. We may assume that $|f|$, and $|m|$ are bounded by 2^k , and therefore the size of a signature in the SNARK-based construction is polynomial in k , and independent of $|f|$, $|m|$, (and even $|f(m)|$).

3.2.3 NIZK-based construction

If one wishes to avoid SNARK-type assumptions, one can obtain a functional signature scheme satisfying both unforgeability and function privacy (but not succinctness) under the more general assumption of standard non-interactive zero-knowledge arguments of knowledge (NIZKAoK). This can be done by essentially replacing the ZK-SNARKs in the construction of the previous section with NIZKAoKs. We remark that our construction hides the function f , but it reveals the size of a circuit computing f .³

Let $(\text{FS3.Setup}, \text{FS3.Keygen}, \text{FS3.Sign}, \text{FS3.Verify})$ be a functional signature scheme which is identical to our previous construction FS2, except that we use a NIZKAoK Π' , instead of the zero-knowledge SNARK system Π .

Theorem 3.10. *If $(\text{Sig.Setup}, \text{Sig.Sign}, \text{Sig.Verify})$ is an existentially unforgeable signature scheme, and Π' is a NIZKAoK, our new functional signature construction $(\text{FS3.Setup}, \text{FS3.Keygen}, \text{FS3.Sign}, \text{FS3.Verify})$ satisfies both unforgeability and function privacy.*

We can use the proof from the previous section, since a zero-knowledge SNARK and a NIZK satisfy the same adaptive zero-knowledge and extractability properties that are used in the proof. The only difference is that a SNARK has a more efficient verification algorithm, and shorter proofs, while a NIZK can be constructed under more general assumptions.

4 Applications of Functional Signatures

In this section we discuss applications of functional signatures to other cryptographic problems, such as constructing delegation schemes and succinct non-interactive arguments.

4.1 SNARGs from Functional Signatures

Recall that in a SNARG protocol for a language L , there is a verifier V , and a prover P who wishes to convince the verifier that an input x is in L . To achieve succinctness, proofs produced by the prover must be sublinear in the size of the input plus the size of the witness.

We show how to use a functional signature scheme supporting keys for functions f describable as polynomial-size circuits, and which has short signatures (i.e of size $r(k) \cdot (|f(m)| + |m|)^{o(1)}$ for a polynomial $r(\cdot)$) to construct a SNARG scheme with preprocessing for any language $L \in NP$ with proof size bounded by $r(k) \cdot (|w| + |x|)^{o(1)}$, where w is the witness and x is the instance. We note that this is the proof size used in the lower bound of [GW11].

Let L be an NP complete language, and R the corresponding relation. The main idea in the construction is for the verifier (or CRS setup) to give out a single signing key for a function whose range consists of exactly those strings that are in L . Note that this can be efficiently described by use of the relation R (where the function also takes as input a witness). Then, with sk_f for this appropriate function f , the prover will

³This is not a concern in the SNARK-base construction, since the size of the signature was independent of the function size.

be able to sign *only* those messages that are in the language L , and hence can use a signature on x as a convincing proof that $x \in L$. The resulting proof is succinct and publicly verifiable.

More explicitly, let $\text{FS} = (\text{FS.Setup}, \text{FS.KeyGen}, \text{FS.Sign}, \text{FS.Verify})$ be a succinct functional signature scheme (as in Definition 3.1) supporting the class \mathcal{F} of polynomial-size circuits. We construct the desired SNARG system $\Pi = (\Pi.\text{Gen}, \Pi.\text{Prove}, \Pi.\text{Verify})$ for NP language L with relation R , as follows:

- $\Pi.\text{Gen}(1^k)$:
 - run the setup for the functional signature scheme, and get $(\text{mvk}, \text{msk}) \leftarrow \text{FS.Setup}(1^k)$
 - generate a signing key $\text{sk}_f \leftarrow \text{FS.KeyGen}(\text{msk}, f)$ where f is the following function:

$$f(x|w) := \begin{cases} x & \text{if } R(x, w) = 1 \\ \perp & \text{otherwise} \end{cases}.$$
 - output $\text{crs} = (\text{mvk}, \text{sk}_f)$
- $\Pi.\text{Prove}(x, w, \text{crs})$:
 - output $\text{FS.Sign}(f, \text{sk}_f, x|w)$
- $\Pi.\text{Verify}(\text{crs}, x, \pi)$:
 - output $\text{FS.Verify}(\text{mvk}, x, \pi)$

Theorem 4.1. *If FS is a functional signature scheme supporting the class \mathcal{F} of polynomial-sized circuits, then Π is a succinct non-interactive argument (SNARG) for NP language L .*

Proof. We address the correctness, soundness, and succinctness of the scheme.

Correctness

The correctness property of the SNARG scheme follows immediately from correctness property of the functional signature scheme. Namely, let R be the relation corresponding to the language L . Then $\forall (x, w) \in R, \forall \text{crs} = (\text{mvk}, \text{sk}_f)$, where $(\text{msk}, \text{mvk}) \leftarrow \text{FS.Setup}(1^k)$, and $\text{sk}_f \leftarrow \text{FS.KeyGen}(\text{msk}, f)$, and $\forall \pi = \sigma$, where $(x, \sigma) \leftarrow \text{FS.Sign}(f, \text{sk}_f, (x, w))$,

$$\Pi.\text{Verify}(\text{crs}, x, \pi) = \text{FS.Verify}(\text{mvk}, x, \sigma) \rightarrow 1.$$

Soundness

The soundness of the proof system follows from the unforgeability property of the signature scheme: since the prover is not given keys for any function except f , he can only sign messages x that are in the range of f , and therefore instances in the language L .

Suppose there exists a PPT adversary Adv for which $\Pr[\text{crs} \leftarrow \Pi.\text{Gen}(1^k); (x, \pi) \leftarrow \text{Adv}(\text{crs}) : x \notin L \wedge \Pi.\text{Verify}(\text{crs}, x, \pi) = 1] = \epsilon(|x|)$, for a non-negligible function $\epsilon(\cdot)$.

Then we can construct an adversary A_{FS} who breaks the unforgeability of the underlying functional signature scheme. A_{FS} gives $\text{crs} = (\text{mvk}, \text{sk}_f)$ to Adv , where mvk is his challenge verification key, and sk_f is the signing key for the function f defined above, which he gets from his key generation oracle.

Adv outputs (x, π) , and A_{FS} uses them as his forgery in the functional signature game. If $x \notin L$, x must not be in the range of L , and therefore (x, π) is a valid forgery. So A_{FS} wins the unforgeability game with probability $\epsilon(|x|)$, which we have assumed is non-negligible.

Succinctness

The size of a proof is equal to the size of a signature in the functional signature scheme, which by assumption is $r(k) \cdot (|f(m)| + |m|)^{o(1)} = r(k) \cdot (|x| + |w|)^{o(1)}$ for a polynomial $r(\cdot)$. \square

Remark 4.2 (Functional PRFs as Functional MACs). Note that functional pseudorandom functions directly imply a notion of functional message authentication codes (MACs), where the master PRF seed s serves as the (shared) master secret MAC key, and a functional PRF subkey sk_f enables one to both MAC and verify messages $f(m)$. Using the transformation above with such a functional MAC in the place of functional signatures yields a *privately verifiable* SNARG system.

Remark 4.3 (Lower bound of [GW11]). Gentry and Wichs showed in [GW11] that SNARG schemes for NP with proof size $r(k) \cdot (|x| + |w|)^{o(1)}$ for polynomial $r(\cdot)$ cannot be obtained using black-box reductions to falsifiable assumptions [Nao03]. Therefore, combined with Theorem 4.1, it follows that in order to obtain a functional signature scheme with signature size $r(k) \cdot (|f(m)| + |m|)^{o(1)}$ we must either rely on non-falsifiable assumptions (as in our SNARK-based construction) or make use of non black-box techniques.

4.2 Connection between functional signatures and delegation

Recall that a delegation scheme allows a client to outsource the evaluation of a function f to a server, while allowing the client to verify the correctness of the computation. The verification process should be more efficient than computing the function. See Definition 2.12 for the required correctness and security properties.

Given a functional signature scheme with signature size $\delta(k)$, and verification time $t(k)$ (which we assume is independent of the size of a function f used in the signing process), we can get a delegation scheme in the preprocessing model with proof size $\delta(k)$ and verification time $t(k)$. Here k is the security parameter.

Let $(\text{FS.Setup}, \text{FS.Prove}, \text{FS.Sign}, \text{FS.Verify})$ be a functional signature scheme supporting the class \mathcal{F} of polynomial-sized circuits. We construct a delegation scheme $(\text{KeyGen}, \text{Encode}, \text{Compute}, \text{Verify})$ as follows:

- $\text{KeyGen}(1^k, f)$:
 - run the setup for the functional signature scheme and generate $(\text{mvm}, \text{msk}) \leftarrow \text{FS.Setup}(1^k)$.
 - define the function $f'(x) := (x, f(x))$, and generate a signing key for f' : $\text{sk}_{f'} \leftarrow \text{FS.KeyGen}(\text{msk}, f')$.
 - output $\text{enc} = \perp$, $\text{evk} = \text{sk}_{f'}$, $\text{vk} = \text{mvk}$.
- $\text{Encode}(\text{enc}, x) = x$: no processing needs to be done on the input.
- $\text{Compute}(\text{evk}, \sigma_x)$:
 - let $\text{sk}_{f'} = \text{evk}$, $x = \sigma_x$
 - generate a signature of $(x, f(x))$ using key $\text{sk}_{f'}$: i.e., $\sigma \leftarrow \text{FS.Sign}(\text{sk}_{f'}, f', x)$
 - output $(f(x), \pi = \sigma)$
- $\text{Verify}(\text{vk}, x, y, \pi_y)$:
 - output $\text{FS.Verify}(\text{vk}, y, \pi_y)$

Theorem 4.4. *If FS is a functional signature scheme supporting the class \mathcal{F} of polynomial-sized circuits, then $(\text{KeyGen}, \text{Encode}, \text{Compute}, \text{Verify})$ is a delegation scheme.*

Correctness

The correctness of the delegation scheme follows from the correctness of the functional signature scheme.

Authenticity

By the unforgeability property of the functional signature scheme, any PPT server will only be able to produce a signature of (x, y) that is in the range of f' : that is, if $y = f(x)$. Thus the server will not be able to sign a pair (x, y) with non-negligible probability, unless $y = f(x)$.

Efficiency

The runtime of the verification algorithm of the delegation scheme is the runtime of the verification algorithm for the signature scheme, $t(k)$. The proof size is equal to the size of a signature in the functional signature scheme, $\delta(k)$.

5 Functional Pseudorandom Functions

In this section we present a formal definition of functional pseudorandom functions (F-PRF), pseudorandom functions with selective access (PRF-SA), and hierarchical functional pseudorandom functions. We present a construction of a functional pseudorandom function family supporting the class of *prefix-fixing functions* based on one-way functions, making use of the Goldreich-Goldwasser-Micali (GGM) tree-based PRF construction [GGM86]. Our construction directly yields a PRF with selective access, and additionally supports hierarchical key generation.

5.1 Definition of Functional PRF

In a standard pseudorandom function family, the ability to evaluate the chosen function is all-or-nothing: a party who holds the secret seed s can compute $F_s(x)$ on all inputs x , whereas a party without knowledge of s cannot distinguish evaluations $F_s(x)$ on requested inputs x from random. We propose the notion of a *functional pseudorandom function (F-PRF)* family, which partly fills this gap between evaluation powers. The idea is that, in addition to a master secret key that can be used to evaluate the pseudorandom function F on any point in the domain, there are additional *secret keys per function* f , which allow one to evaluate F on y for any y for which there exists an x such that $f(x) = y$ (i.e., y is in the range of f).

Definition 5.1 (Functional PRF). We say that a PRF family $\mathcal{F} = \{F_s : D \rightarrow R\}_{s \in S}$ is a *functional pseudorandom function (F-PRF)* if there exist additional algorithms

$\text{KeyGen}(s, f)$: On input a seed $s \in S$ and function description $f : A \rightarrow D$ from some domain A to D , the algorithm KeyGen outputs a key sk_f .

$\text{Eval}(\text{sk}_f, f, x)$: On input key sk_f , function $f : A \rightarrow D$, and input $x \in A$, then Eval outputs the PRF evaluation $F_s(f(x))$.

which satisfy the following properties:

- **Correctness:** For every (efficiently computable) function $f : A \rightarrow D$, $\forall x \in A$, it holds that

$$\forall s \leftarrow S, \forall \text{sk}_f \leftarrow \text{KeyGen}(s, f), \text{Eval}(\text{sk}_f, f, x) = F_s(f(x)).$$

- **Pseudorandomness:** Given a set of keys $\text{sk}_{f_1} \dots \text{sk}_{f_l}$ for functions $f_1 \dots f_l$, the evaluation of $F_s(y)$ should remain pseudorandom on all inputs y that are not in the range of any of the functions $f_1 \dots f_l$. That is, for any PPT adversary \mathcal{A} , the advantage of \mathcal{A} in distinguishing between the following two experiments is negligible (for any polynomial $l = l(k)$):

Experiment Rand	Experiment PRand
<u>Key query Phase</u>	<u>Key query Phase</u>
$(\text{pp}, s) \leftarrow \text{Gen}(1^k)$	$(\text{pp}, s) \leftarrow \text{Gen}(1^k)$
$f_1 \leftarrow \mathcal{A}(\text{pp})$	$f_1 \leftarrow \mathcal{A}(\text{pp})$
$\text{sk}_{f_1} \leftarrow \text{KeyGen}(s, f_1)$	$\text{sk}_{f_1} \leftarrow \text{KeyGen}(s, f_1)$
\vdots	\vdots
$f_l \leftarrow \mathcal{A}(\text{pp}, f_1, \text{sk}_{f_1}, \dots, f_{l-1}, \text{sk}_{f_{l-1}})$	$f_l \leftarrow \mathcal{A}(\text{pp}, f_1, \text{sk}_{f_1}, \dots, f_{l-1}, \text{sk}_{f_{l-1}})$
$\text{sk}_{f_l} \leftarrow \text{KeyGen}(s, f_l)$	$\text{sk}_{f_l} \leftarrow \text{KeyGen}(s, f_l)$
<u>Challenge Phase</u>	<u>Challenge Phase</u>
$H \leftarrow \mathbb{F}_{D \rightarrow R}$ a random function	
$b \leftarrow \mathcal{A}^{\mathcal{O}_{s,H}^{\{f_i\}}(\cdot)}(f_1, \text{sk}_{f_1}, \dots, f_l, \text{sk}_{f_l})$	$b \leftarrow \mathcal{A}^{F_s(\cdot)}(f_1, \text{sk}_{f_1}, \dots, f_l, \text{sk}_{f_l})$

where $\mathcal{O}_{s,H}^{\{f_i\}}(y) := \begin{cases} F_s(y) & \text{if } \exists i \in [l] \text{ and } x \text{ s.t. } f_i(x) = y \\ H(y) & \text{otherwise} \end{cases}$.

Note that, as defined, the oracle $\mathcal{O}_{s,H}^{\{f_i\}}(y)$ need not be efficiently computable. This inefficiency stems both from sampling a truly random function H , and from testing whether the adversary's evaluation queries y are contained within the range of one of his previously queried functions f_i . However, within particular applications, the system can be set up so that this oracle is efficiently simulatable: For example, evaluations of a truly random function can be simulated by choosing each queried evaluation one at a time; Further, the range of the relevant functions f_i may be efficiently testable given trapdoor information (e.g., determining the range of $f : r \mapsto \text{Enc}(\text{pk}, 0; r)$ for a public-key encryption scheme is infeasible given only pk but efficiently testable given the secret key).

We also consider a weaker security definition, where the adversary has to reveal which functions he will request keys for before seeing the public parameters or any of the keys. We refer to this as *selective pseudorandomness*.

Definition 5.2 (Selectively Secure F-PRF). We say a PRF family is a *selectively secure* functional pseudorandom function if the algorithms $\text{KeyGen}, \text{Eval}$ satisfy the correctness property above, and the following selective pseudorandomness property.

- **Selective Pseudorandomness:** For any PPT adversary \mathcal{A} , the advantage of \mathcal{A} in distinguishing between the following two experiments is negligible:

Experiment Sel-Rand	Experiment Sel-PRand
<u>Key query Phase</u>	<u>Key query Phase</u>
$f_1, \dots, f_l \leftarrow \mathcal{A}$	$f_1, \dots, f_l \leftarrow \mathcal{A}$
$(\text{pp}, s) \leftarrow \text{Gen}(1^k)$	$(\text{pp}, s) \leftarrow \text{Gen}(1^k)$
$\text{sk}_{f_1} \dots \text{sk}_{f_l} \leftarrow \text{KeyGen}(s, f_1, \dots, f_l)$	$\text{sk}_{f_1} \dots \text{sk}_{f_l} \leftarrow \text{KeyGen}(s, f_1, \dots, f_l)$
<u>Challenge Phase</u>	<u>Challenge Phase</u>
$H \leftarrow \mathbb{F}_{D \rightarrow R}$ a random function	
$b \leftarrow \mathcal{A}^{\mathcal{O}_{s,H}^{\{f_i\}}(\cdot)}(f_1, \text{sk}_{f_1}, \dots, f_l, \text{sk}_{f_l})$	$b \leftarrow \mathcal{A}^{F_s(\cdot)}(f_1, \text{sk}_{f_1}, \dots, f_l, \text{sk}_{f_l})$

$$\text{where } \mathcal{O}_{s,H}^{\{f_i\}}(y) := \begin{cases} F_s(y) & \text{if } \exists i \in [l] \text{ and } x \text{ s. t. } f_i(x) = y \\ H(y) & \text{otherwise} \end{cases}.$$

A special case of functional PRFs are when access control is to be determined by predicates. (Indeed, fitting within the F-PRF framework, one can emulate predicate policies by considering the corresponding functions $f_P(x) = x$ if $P(x) = 1$ and $= \perp$ if $P(x) = 0$). For completeness, we now present the corresponding formal definition, which we refer to as PRFs with *selective access*.

Definition 5.3 (PRF with Selective Access). We say that a PRF family $\mathcal{F} = \{F_s : D \rightarrow R\}_{s \in S}$ is a *pseudorandom function family with selective access (PRF-SA)* for a class of predicates \mathcal{P} on D if there exist additional efficient algorithms

$\text{KeyGen}(s, P)$: On input a seed $s \in S$ and predicate $P \in \mathcal{P}$, KeyGen outputs a key sk_P .

$\text{Eval}(\text{sk}_P, P, x)$: On input key sk_P and input $x \in D$, if it holds that $P(x) = 1$ then Eval outputs the PRF evaluation $F_s(x)$.

which satisfy the following properties:

- **Correctness:** For each predicate $P \in \mathcal{P}$, $\forall x \in D$ s.t. $P(x) = 1$, it holds that

$$\forall s \leftarrow S, \forall \text{sk}_P \leftarrow \text{KeyGen}(s, P), \text{Eval}(\text{sk}_P, P, x) = F_s(x)$$

- **Pseudorandomness:** Given a set of keys $\text{sk}_{P_1} \dots \text{sk}_{P_l}$ for predicate $P_1 \dots P_l$, the evaluation of $F_s(x)$ should remain pseudorandom on all inputs x for which $P_1(x) = 0 \wedge \dots \wedge P_l(x) = 0$. That is, for any PPT adversary \mathcal{A} , the advantage of \mathcal{A} in distinguishing between the following two experiments is negligible:

Experiment Rand	Experiment PRand
Query Phase	Query Phase
$(pp, s) \leftarrow \text{Gen}(1^k)$	$(pp, s) \leftarrow \text{Gen}(1^k)$
$P_1 \leftarrow \mathcal{A}(pp)$	$P_1 \leftarrow \mathcal{A}(pp)$
$sk_{P_1} \leftarrow \text{KeyGen}(s, P_1)$	$sk_{P_1} \leftarrow \text{KeyGen}(s, P_1)$
\vdots	\vdots
$P_l \leftarrow \mathcal{A}(pp, P_1, sk_{P_1} \dots P_{l-1}, sk_{P_{l-1}})$	$P_l \leftarrow \mathcal{A}(pp, P_1, sk_{P_1} \dots P_{l-1}, sk_{P_{l-1}})$
$sk_{P_l} \leftarrow \text{KeyGen}(s, P_l)$	$sk_{P_l} \leftarrow \text{KeyGen}(s, P_l)$
Challenge Phase	Challenge Phase
$H \leftarrow \mathbb{F}_{D \rightarrow R}$ a random function	
$b \leftarrow \mathcal{A}^{\mathcal{O}_{s,H}^P}(\cdot)(P_1, sk_{P_1}, \dots, P_l, sk_{P_l})$	$b \leftarrow \mathcal{A}^{F_s(\cdot)}(P_1, sk_{P_1}, \dots, P_l, sk_{P_l})$

$$\text{where } \mathcal{O}_{s,H}^P(x) := \begin{cases} F_s(x) & \text{if } \exists i \in [l], P_i(x) = 1 \\ H(x) & \text{otherwise} \end{cases}.$$

Finally, we consider *hierarchical* F-PRFs, where a party holding key sk_f for function $f : B \rightarrow D$ can generate a subsidiary key $sk_{f \circ g}$ for a second function $g : A \rightarrow B$.

Definition 5.4 (Hierarchical F-PRF). We say that an F-PRF family $(\{F_s\}_s, \text{KeyGen}, \text{Eval})$ is *hierarchical* if the algorithm KeyGen is replaced by a more general algorithm:

$\text{SubkeyGen}(sk_f, g)$: On input a functional secret key sk_f for function $f : B \rightarrow C$ (where the master secret key is considered to be sk_1 for the identity function $f(x) = x$), and function description $g : A \rightarrow B$ for some domain A , SubkeyGen outputs a secret subkey $sk_{f \circ g}$ for the composition $f \circ g$.

satisfying the following properties:

- **Correctness:** Any key sk_g generated via a sequence of SubkeyGen executions will correctly evaluate $F_s(f(x))$ on each value y for which they know a preimage x with $g(x) = y$. Formally, for every sequence of (efficiently computable) functions f_1, \dots, f_ℓ with $f_i : A_i \rightarrow A_{i-1}$, $\forall y \in A_0$ s.t. $\exists x \in A_\ell$ for which $f_1 \circ \dots \circ f_\ell(x) = y$, it holds that

$$\forall sk_1 \leftarrow S, \quad \forall sk_{f_1 \circ \dots \circ f_i} \leftarrow \text{SubkeyGen}(sk_{f_1 \circ \dots \circ f_{i-1}}, f_i) \text{ for } i = 0, \dots, \ell,$$

$$\text{Eval}(sk_{f_1 \circ \dots \circ f_\ell}, (f_1 \circ \dots \circ f_\ell), x) = F_{sk_1}(y).$$

- **Pseudorandomness:** The pseudorandomness property of Definition 5.1 holds, with the slight modification that the adversary may adaptively make queries of the following kind, corresponding to receiving subkeys sk_g generated from unknown functional keys sk_f . The query phase begins with a master secret key $s \leftarrow S$ being sampled and assigned identity $id = 1$. Loosely, GenerateKey generates a new subkey of an existing (possibly unknown) key indexed by id , and keeps the resulting key hidden. RevealKey simply reveals the generated key indexed by id .

$\text{GenerateKey}(id, g)$: If no key exists with identity id then output \perp and terminate; otherwise denote this key by sk_f . The challenger generates a g -subkey from sk_f as $sk_{f \circ g} \leftarrow \text{SubkeyGen}(sk_f, g)$, and assigns this key a unique identity id' . The new value id' is output, and the resulting key $sk_{f \circ g}$ is kept secret.

$\text{RevealKey}(id)$: If no key exists with identity id then output \perp and terminate; otherwise output the corresponding key sk_f .

In the challenge phase, the adversary's evaluation queries are answered either (1) consistently pseudorandom, or (2) pseudorandom for all inputs y for which the adversary was given a key sk_f in a RevealKey query with $y \in \text{Range}(f)$, and random for all other inputs.

5.2 Construction Based on OWF

We now construct a functional pseudorandom function family $F_s : \{0,1\}^n \rightarrow \{0,1\}^n$ supporting the class of prefix-fixing functions, based on the Goldreich-Goldwasser-Micali (GGM) tree-based PRF construction [GGM86]. More precisely, our construction supports the function class

$$\mathcal{F}_{\text{pre}} = \left\{ f_z(x) : \{0,1\}^n \rightarrow \{0,1\}^n \mid z \in \{0,1\}^m \text{ for } m \leq n \right\},$$

$$\text{where } f_z(x) := \begin{cases} x & \text{if } (x_1 = z_1) \wedge \cdots \wedge (x_m = z_m) \\ \perp & \text{otherwise} \end{cases}.$$

Recall that the GGM construction makes use of a length-doubling pseudorandom generator $G : \{0,1\}^k \rightarrow \{0,1\}^{2k}$ (which can be constructed from any one-way function). Denoting the two halves of the output of G as $G(y) = G_0(y)G_1(y)$, the PRF with seed s is defined as $F_s(y) = G_{y_k}(\cdots G_{y_2}(G_{y_1}(s)))$.

We show that we can obtain a functional PRF for \mathcal{F}_{pre} by adding the following two algorithms on top of the GGM PRF construction. Intuitively, in these algorithms the functional secret key sk_{f_z} corresponding to a queried function $f_z \in \mathcal{F}_{\text{pre}}$ will be the partial evaluation of the GGM prefix corresponding to prefix z : i.e., the label of the node corresponding to node z in the GGM evaluation tree. Given this partial evaluation, a party will be able to compute the completion for any input x which has z as a prefix. However, as we will argue, the evaluation on all other inputs will remain pseudorandom.

KeyGen(s, f_z) : output $G_{z_m}(\cdots G_{z_2}(G_{z_1}(s)))$, where $m = |z|$

Eval(sk_{f_z}, y) : output $\begin{cases} G_{y_n}(\cdots G_{y_{m+2}}(G_{y_{m+1}}(sk_{f_z}))) & \text{if } y_1 = z_1 \wedge \cdots \wedge y_m = z_m \\ \perp & \text{otherwise} \end{cases}$

We first prove that this construction yields an F-PRF with selective security (i.e., when the adversary's key queries are specified a priori). We then present a sequence of corollaries for achieving full security, PRFs with selective access, and hierarchical F-PRFs. We also focus on the specific application of *punctured* PRFs [SW13].

Theorem 5.5. *Based on the existence of one-way functions, the GGM pseudorandom function family together with algorithms KeyGen and Eval defined as above, is a selectively secure functional PRF for the class of functions \mathcal{F}_{pre} , as per Definition 5.2.*

Proof. We will reduce the pseudorandom property of our functional PRF scheme to the security of the underlying PRG. Recall that (as per Definition 5.2), the functional PRF requires indistinguishability of experiments Sel-PRand and Sel-Rand, in which the adversary makes key queries (which are answered honestly), and then makes evaluation queries, which are either answered consistently (PRand) or randomly (Rand). At a high level, we will show that both Experiment Sel-Rand and Experiment Sel-PRand are indistinguishable from a third experiment where, in the query phase, the adversary's queries are answered randomly (except when one query is a prefix of another, in which case we need to ensure consistency), and in the challenge phase the adversary's queries are answered randomly. Both claims will be proved using a hybrid argument similar to the proof of the original GGM construction.

Let $f_1, \dots, f_l \in \mathcal{F}_{\text{pre}}$ be the functions queried by the adversary. Let P_1, \dots, P_l be the corresponding prefixes. We consider the following experiments:

- Exp 1.** Experiment Sel-PRand. In the key query phase, the key for each function f_i corresponding to prefix P_i is obtained (honestly) by following the corresponding path in the GGM tree. In the challenge phase, the adversary's evaluation queries are answered (honestly) with the corresponding pseudorandom values. We denote the probability that an adversary Adv outputs 1 in this experiment by $\text{output}_{\text{Exp1}}^{\text{Adv}}$.
- Exp 2.** Keys for the queried functions $f_1, \dots, f_l \in \mathcal{F}_{\text{pre}}$ corresponding to prefixes P_i are computed randomly, up to consistency among queried sub-prefixes. This takes place as follows (recall that all queries are made up front):

- for each f_i , if no prefix of P_i is also queried by the adversary in his keygen queries, then sk_{f_i} is assigned a random value.
- otherwise, let P_j be the shortest such prefix that is also queried (so that sk_{f_j} has already been defined by the previous case). Then sk_{f_i} is computed by honestly applying to sk_{f_j} the sequence of PRG's determined by the bits of P_i following P_j .

In the challenge phase, the adversary's evaluation queries are answered with random values. If a query is repeated, we answer consistently. We denote the probability that an adversary Adv outputs 1 in this experiment by $\text{output}_{\text{Exp2}}^{\text{Adv}}$.

Exp 3 Experiment Sel-Rand. In the key query phase, the key for each function f_i corresponding to prefix P_i is obtained (honestly) by following the corresponding path in GGM tree, and. In the challenge phase, the adversary's evaluation queries (to values not computable by himself already) are answered with random values. If a query is repeated, we answer consistently. We denote the probability that an adversary Adv outputs 1 in this experiment by $\text{output}_{\text{Exp3}}^{\text{Adv}}$.

Note that that experiment described in Exp 1 is Experiment Sel-PRand in the Functional PRF definition, and the experiment described in Exp 3 is Experiment Sel-Rand.

Lemma 5.6. *For any PPT adversary Adv*

$$|\text{output}_{\text{Exp1}}^{\text{Adv}} - \text{output}_{\text{Exp2}}^{\text{Adv}}| = \text{negl}(n).$$

Proof. Suppose there exists an adversary Adv , such that $|\text{output}_{\text{Exp1}}^{\text{Adv}} - \text{output}_{\text{Exp2}}^{\text{Adv}}| = \epsilon(n)$ for some non-negligible $\epsilon(n)$. Wlog, assume that $\text{output}_{\text{Exp2}}^{\text{Adv}} - \text{output}_{\text{Exp1}}^{\text{Adv}} = \epsilon(n) > 0$. We claim that we can use Adv to construct an adversary A_{PRG} that breaks the security of the underlying pseudorandom generator. Recall in the PRG challenge, A_{PRG} receives a polynomial-sized set of values, which are either random or random outputs of the PRG.

We use a hybrid argument, and define Exp^i for $i \in [n]$. The value i corresponds to the level of the tree where A_{PRG} will place his challenge values when interacting with Adv .

In Exp^i , in the key query phase, the key for each function f_j corresponding to prefix P_j of length $|P_j| = m$ is computed as follows:

- if no other queried prefix is a prefix of P_j and $m \leq i$, return a random string of size n .
- if no other queried prefix is a prefix of P_j and $m > i$, set the label of P_j 's ancestor on the i^{th} level to a randomly sampled n -bit string, and then apply the pseudorandom generators to it as in the GGM construction according to the remaining bits of P_j until the m^{th} level, and return the resulting string of size n .
- if some other queried prefix is a prefix of P_j , let sk_{f_h} be the key corresponding to the shortest such queried prefix P_h . To obtain the key for P_j , apply the pseudorandom generators to sk_{f_h} as in the GGM construction according to the remaining bits of P_j , up to the m^{th} level of the tree.

In the challenge phase, the answers to the adversary's evaluation queries x are computed as follows:

- let $x^{(i)}$ denote the i -bit prefix of the queried input x . If the node corresponding to $x^{(i)}$ in the tree has not yet been labeled, then a random value is chosen and set as this label. The response to the adversary's query is then computed by applying the PRGs to the label, as determined by the $(i+1)$ to n bits of the queried input x .

Since $\text{output}_{\text{Exp2}}^{\text{Adv}} - \text{output}_{\text{Exp1}}^{\text{Adv}} = \epsilon(n)$, there must exist an i such that:

$$\Pr[\text{Adv} \rightarrow 1 \text{ in } \text{Exp}^i] - \Pr[\text{Adv} \rightarrow 1 \text{ in } \text{Exp}^{i+1}] \geq \frac{\epsilon(n)}{n}.$$

Our constructed PRG adversary A_{PRG} plays the role of the challenger in the game with Adv , chooses a random $i \in [n]$ and places his PRG challenges there. That is, in the key query phase, A_{PRG} computes the keys for functions f_i corresponding to prefix P_j , of length $|P_j| = m$ as follows:

- if no other queried prefix is a prefix of P_j and $m < i$, return a random string of size n .
- if no other queried prefix is a prefix of P_j and $m = i$, return one of A_{PRG} 's challenge values.
- if no other queried prefix is a prefix of P_j and $m > i$, set a challenge string as the ancestor of P_j on the i^{th} level, and then apply the pseudorandom generators to it as in the GGM construction until the m^{th} level and return the resulting string of size n .
- if some other queried prefix is a prefix of P_j , let sk_{f_h} be the key corresponding to the shortest such queried prefix, P_h . To obtain the key for P_j , apply the pseudorandom generators to sk_{f_h} as in the GGM construction, up to the m^{th} level of the tree.

In the challenge phase, the answers to the adversary's evaluation queries x are computed as follows:

- let $x^{(i)}$ denote the i -bit prefix of the queried input x . If the node corresponding to $x^{(i)}$ in the tree has not yet been labeled, then one of A_{PRG} 's challenge values is chosen and set as the label. The response to the adversary's query is then computed by applying the PRGs to the label, as determined by the $(i + 1)$ to n bits of the queried input x .

Comparing the experiment above to Exp^i and Exp^{i+1} , we can see that, if the inputs to A_{PRG} are random, A_{PRG} behaves as the challenger in Exp^i , and if they are the output of a PRG, he behaves as the challenger in Exp^{i+1} .

At the end A_{PRG} outputs the same answer as Adv in its own security game.

$$\begin{aligned}
& \Pr[A_{PRG} \text{ guesses correctly}] \\
&= \frac{1}{2} \Pr[A_{PRG} \rightarrow 1 | \text{challenge values random}] + \frac{1}{2} \Pr[A_{PRG} \rightarrow 0 | \text{challenge values are output of a PRG}] \\
&= \frac{1}{2} \Pr[Adv \text{ outputs 1 in } Exp^i] + \frac{1}{2} \Pr[Adv \text{ outputs 0 in } Exp^{i+1}] \\
&= \frac{1}{2} \Pr[Adv \text{ outputs 1 in } Exp^i] + \frac{1}{2} (1 - \Pr[Adv \text{ outputs 1 in } Exp^{i+1}]) \\
&= \frac{1}{2} + \frac{1}{2} (\Pr[Adv \text{ outputs 1 in } Exp^i] - \Pr[Adv \text{ outputs 1 in } Exp^{i+1}]) \\
&\geq \frac{1}{2} + \frac{\epsilon(n)}{2n}
\end{aligned}$$

If $\epsilon(n)$ is non-negligible, A_{PRG} can distinguish between random values and outputs of a pseudorandom generator with non-negligible advantage, which would break the security of the underlying pseudorandom generator. This completes the proof of the lemma. \square

Lemma 5.7. *For any PPT adversary Adv*

$$|\text{output}_{Exp2}^{Adv} - \text{output}_{Exp3}^{Adv}| = \text{negl}(n).$$

Proof. We use a similar hybrid argument: In Exp^i , in the key query phase, the key for the functions corresponding to prefix P_j , of length $|P_j| = m$ is computed as before:

- if no other queried prefix is a prefix of P_j and $m \leq i$, return a random string of size n .
- if no other queried prefix is a prefix of P_j and $m > i$, set a random string as the parent of P_j on the i^{th} level, and then apply the pseudorandom generators to it as in the GGM construction until the m^{th} level and return the resulting string of size n .
- if some other queried prefix is a prefix of P_j , let sk_{f_h} be the key corresponding to the shortest queried prefix of P_j , P_h . To obtain the key for P_j , apply the pseudorandom generators to sk_{f_h} as in the GGM construction, up to the m^{th} level of the tree.

In the challenge phase, the adversary's queries are answered with random values, unless he has already received a key that allows him to compute the PRF on his queried value, in which case the query is answered consistently.

The first hybrid, Exp^0 , is Exp 3, and the last hybrid, Exp^n is Exp 2. \square

From the previous lemmas, we can conclude that, for any PPT adversary Adv

$$|\text{output}_{Exp1}^{Adv} - \text{output}_{Exp3}^{Adv}| = \text{negl}(n).$$

This is equivalent to saying that no PPT adversary can distinguish between Experiment Sel-PRand and Experiment Sel-Rand in the Functional PRF definition. That is, the construction is a secure F-PRF. \square

Remark 5.8. We remark that one can directly obtain a *fully* secure F-PRF for \mathcal{F}_{pre} (as in Definition 5.1) from our selectively secure construction, with a loss of $\frac{1}{2^n}$ in security for each functional secret key sk_{f_z} queried by the adversary. This is achieved simply by guessing the adversary's query $f_z \in \mathcal{F}_{\text{pre}}$. For appropriate choices of input size n and security parameter k , this can still provide meaningful security.

As an immediate corollary of Theorem 5.5, we obtain a (selectively secure) PRF with selective access for the class of equivalent prefix-matching predicates $\mathcal{P}_{\text{pre}} = \{P_z : \{0,1\}^n \rightarrow \{0,1\} | z \in \{0,1\}^m \text{ for } m \leq n\}$, where $P_z(x) := 1$ if $(x_1 = z_1) \wedge \dots \wedge (x_m = z_m)$ and 0 otherwise.

Corollary 5.9. *Based on the existence of one-way functions, the GGM pseudorandom function family together with algorithms KeyGen and Eval defined as above, is a selectively secure functional PRF for the class of predicates \mathcal{P}_{pre} .*

Our F-PRF construction has the additional benefit of being *hierarchical*. That is, given a secret key sk_{f_z} for a prefix $z \in \{0,1\}^m$, a party can generate subordinate secret keys $\text{sk}_{f_{z'}}$ for any $z' \in \{0,1\}^{m'}$, $m' > m$ agreeing with z on the first m bits. This secondary key generation process is accomplished simply by applying the PRGs to sk_{f_z} , traversing the GGM tree according to the additional bits of z' . We thus achieve the following corollary.

Corollary 5.10. *Based on the existence of one-way functions, the GGM pseudorandom function family together with algorithms KeyGen and Eval defined as above, is a (selectively secure) hierarchical functional PRF for the class of predicates \mathcal{P}_{pre} .*

The pseudorandomness property can be proved using the same techniques as in the proof of Theorem 5.5.

5.2.1 Punctured Pseudorandom Functions

Punctured PRFs, formalized by [SW13], are a special case of functional PRFs where one can generate keys for the function family $\mathcal{F} = \{f_x(y) = y \text{ if } y \neq x, \text{ and } \perp \text{ otherwise}\}$. Such PRFs have recently been shown to have important applications, including use as a primary technique in proving security of various cryptographic primitives based on the existence of indistinguishability obfuscation (see, e.g., [SW13, HSW13]).

The existence of a functional PRF for the prefix-fixing function family gives a construction of punctured PRFs. Namely, a punctured key sk_x allowing one to compute the PRF on all inputs except $x = x_1 \dots x_n$ consists of n functional keys for the prefix-fixing function family for prefixes: $(\bar{x}_1), (x_1\bar{x}_2), (x_1x_2\bar{x}_3), \dots, (x_1x_2 \dots x_{n-1}\bar{x}_n)$.

Our GGM-based construction in the previous section thus directly yields a selectively secure punctured PRF based on OWFs.

Corollary 5.11 (Selectively-Secure Punctured PRFs). *Assuming the existence of OWF, there exists a selectively secure punctured PRF for any desired poly-size input length.*

When considering full security, this may seem an inhibiting limitation, as naïve complexity leveraging over each of the n released keys would incur a tremendous loss in security. However, for a punctured PRF, these n keys are not independently chosen: rather, there is a one-to-one correspondence between the input x that is punctured, and corresponding set of n prefix-fixing keys we give out. This means there are only 2^n possible sets of key queries made by a punctured PRF adversary (as opposed to 2^{n^2} possible choices of n independent prefix queries), and thus, in the full-to-selective security reduction, we lose only a factor of 2^{-n} in the security (as the reduction needs only to guess which of these 2^n query sets will be made by the adversary). Given a desired level of security k and input size $n = n(k)$, and assuming an underlying OWF secure against all adversaries that run in time 2^{K^ϵ} when implemented with security parameter K for some constant $0 < \epsilon < 1$, then by setting $K = n^{1/\epsilon}$, we obtain a fully secure puncturable PRF.

Corollary 5.12. *Assuming the existence of 2^{K^ϵ} -hard OWF for security parameter K and some constant $0 < \epsilon$, there exists a (fully) secure punctured PRF for any desired poly-size input length.*

References

- [AGVW13] Shweta Agrawal, Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption: New perspectives and lower bounds. In *CRYPTO*, 2013.
- [BCCT12] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In *ITCS*, pages 326–349, 2012.
- [BCCT13] Nir Bitansky, Ran Canetti, Alessandro Chiesa, and Eran Tromer. Recursive composition and bootstrapping for snarks and proof-carrying data. In *STOC*, pages 111–120, 2013.
- [BF11] Dan Boneh and David Mandell Freeman. Homomorphic signatures for polynomial functions. In *EUROCRYPT*, pages 149–168, 2011.
- [BF13] Mihir Bellare and Georg Fuchsbauer. Policy-based signatures. Cryptology ePrint Archive, Report 2013/413, 2013.
- [BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112, 1988.
- [BG89] Mihir Bellare and Shafi Goldwasser. New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs. In *CRYPTO*, pages 194–211, 1989.
- [BGI⁺01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.
- [BMS13] Michael Backes, Sebastian Meiser, and Dominique Schröder. Delegatable functional signatures. Cryptology ePrint Archive, Report 2013/408, 2013.
- [BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.
- [BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.
- [BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. Cryptology ePrint Archive, Report 2013/352, 2013.
- [DH76] Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.
- [FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317, 1990.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986.

- [GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In *CRYPTO*, pages 465–482, 2010.
- [GK05] Shafi Goldwasser and Yael Tauman Kalai. On the impossibility of obfuscation with auxiliary input. In *FOCS*, pages 553–562, 2005.
- [GKP⁺12] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. Succinct functional encryption and applications: Reusable garbled circuits and beyond. *IACR Cryptology ePrint Archive*, 2012:733, 2012.
- [GKP⁺13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nikolai Zeldovich. Overcoming the worst-case curse for cryptographic constructions. In *CRYPTO*, 2013.
- [GM82] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *STOC*, pages 365–377, 1982.
- [GMR88] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.*, 17(2):281–308, 1988.
- [GMW86] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to prove all np-statements in zero-knowledge, and a methodology of cryptographic protocol design. In *CRYPTO*, pages 171–185, 1986.
- [GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO*, pages 162–179, 2012.
- [GW11] Craig Gentry and Daniel Wichs. Separating succinct non-interactive arguments from all falsifiable assumptions. In *STOC*, pages 99–108, 2011.
- [GW12] Rosario Gennaro and Daniel Wichs. Fully homomorphic message authenticators. *IACR Cryptology ePrint Archive*, 2012:290, 2012.
- [HSW13] Susan Hohenberger, Amit Sahai, and Brent Waters. Replacing a random oracle: Full domain hash from indistinguishability obfuscation. *Cryptology ePrint Archive*, Report 2013/509, 2013.
- [KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. *Cryptology ePrint Archive*, Report 2013/379, 2013.
- [Lys02] Anna Lysyanskaya. Unique signatures and verifiable random functions from the dh-ddh separation. In *CRYPTO*, pages 597–612, 2002.
- [MRV99] Silvio Micali, Michael O. Rabin, and Salil P. Vadhan. Verifiable random functions. In *FOCS*, pages 120–130, 1999.
- [Nao03] Moni Naor. On cryptographic assumptions and challenges. In *CRYPTO*, pages 96–109, 2003.
- [PST13] Charalampos Papamanthou, Elaine Shi, and Roberto Tamassia. Signatures of correct computation. In *TCC*, pages 222–242, 2013.
- [Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394, 1990.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
- [SW13] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: Deniable encryption, and more. *Cryptology ePrint Archive*, Report 2013/454, 2013.

Aggregate Pseudorandom Functions and Connections to Learning

Aloni Cohen*

Shafi Goldwasser[†]

Vinod Vaikuntanathan[‡]

Abstract

In the first part of this work, we introduce a new type of pseudo-random function for which “aggregate queries” over exponential-sized sets can be efficiently answered. We show how to use algebraic properties of underlying classical pseudo random functions, to construct such “aggregate pseudo-random functions” for a number of classes of aggregation queries under cryptographic hardness assumptions. For example, one aggregate query we achieve is the product of all function values accepted by a polynomial-sized read-once boolean formula. On the flip side, we show that certain aggregate queries are impossible to support. Aggregate pseudo-random functions fall within the framework of the work of Goldreich, Goldwasser, and Nussboim [GGN10] on the “Implementation of Huge Random Objects,” providing truthful implementations of pseudo-random functions for which aggregate queries can be answered.

In the second part of this work, we show how various extensions of pseudo-random functions considered recently in the cryptographic literature, yield impossibility results for various extensions of machine learning models, continuing a line of investigation originated by Valiant and Kearns in the 1980s. The extended pseudo-random functions we address include constrained pseudo random functions, aggregatable pseudo random functions, and pseudo random functions secure under related-key attacks.¹

*MIT.

[†]MIT and the Weizmann Institute of Science.

[‡]MIT.

¹©IACR 2015. This article is a minor revision of the version published by Springer-Verlag.

Contents

1	Introduction	1
1.1	Our Results: Aggregate Pseudo Random Functions	2
1.1.1	Related Work to Aggregate PRFs	6
1.2	Our Results: Augmented PRFs and Computational Learning	6
1.2.1	Constrained PRFs and limits on Restriction Access learnability	7
1.2.2	New Learning Models Inspired by the Study of PRFs	8
2	Aggregate PRF	11
2.1	A General Security Theorem for Aggregate PRFs	12
2.2	Impossibility of Aggregate PRF for General Sets	14
3	Constructions of aggregate PRF	14
3.1	Generic Construction for Interval Sets	15
3.2	Bit-Fixing Aggregate PRF from DDH	16
3.2.1	Construction	16
3.3	Decision Trees	17
3.4	Read-once formulas	18
3.4.1	Construction	18
4	Connection to Learning	20
4.1	Preliminaries	20
4.2	Membership queries with restriction access	22
4.2.1	MQ _{RA} learning	22
4.2.2	Constrained PRFs	23
4.2.3	Hardness of restriction access Learning	24
4.3	Learning with related concepts	26
4.3.1	RKA PRFs	27
4.3.2	Hardness of related concept learning	27
4.4	Learning with Aggregate Queries	29
4.4.1	Hardness of aggregate query learning	29
4.4.2	Acknowledgements	30
A	Simple Positive Results	33
A.1	Related-concept	33
A.2	Aggregate queries	35

1 Introduction

Pseudo-random functions (PRF), introduced by Goldreich, Goldwasser and Micali [GGM86], are a family of indexed functions for which there exists a polynomial-time algorithm that, given an index (which can be viewed as a secret key) for a function, can evaluate it, but no probabilistic polynomial-time algorithm without the secret key can distinguish the function from a truly random function – even if allowed oracle query access to the function. Pseudo-random functions have been shown over the years to be useful for numerous cryptographic applications. Interestingly, aside from their cryptographic applications, PRFs have also been used to show impossibility of computational learning in the membership queries model [Val84], and served as the underpinning of the proof of Razborov and Rudich [RR97] that natural proofs would not suffice for unrestricted circuit lower bounds.

Since their inception in the mid eighties, various *augmented* pseudo random functions with *extra* properties have been proposed, enabling more sophisticated forms of access to PRFs and more structured forms of PRFs. This was first done in the work of Goldreich, Goldwasser, and Nussboim [GGN10] on how to efficiently construct “huge objects” (e.g. a large graph implicitly described by access to its adjacency matrix) which maintain combinatorial properties expected of a *random* “huge object.” Furthermore, they show several implementations of varying quality of such objects for which complex global properties can be computed, such as computing cliques in a random graph, computing random function inverses from a point in the range, and computing the parity of a random function’s values over huge sets. More recently, further augmentations of PRFs have been proposed, including: the works on constrained PRFs² [KPTZ13a, BGI14a, BW13a] which can release auxiliary secret keys whose knowledge enables computing the PRF in a restricted number of locations without compromising pseudo-randomness elsewhere; key-homomorphic PRFs [BLMR13] which are homomorphic with respect to the keys; and related-key secure PRFs [BC10, ABPP14]. These constructions yield fundamental objects with often surprising applications to cryptography and elsewhere. A case in point is the truly surprising use of constrained PRFs [SW14], to show that indistinguishability obfuscation can be used to resolve a long-standing problem of deniable encryption, among many others.

In the first part of this paper, we introduce a new type of augmented PRF which we call *aggregate pseudo random functions* (AGG-PRF). An AGG-PRF is a family of indexed functions each associated with a secret key, such that *given the secret key*, one can compute aggregates of the values of the function *over super-polynomially large sets* in *polynomial time*; and yet without the secret key, access to such aggregated values cannot enable a polynomial time adversary (distinguisher) to distinguish the function from random, even when the adversary can make *aggregate queries*. Note that the distinguisher can request and receive an aggregate of the function values over sets (of possibly super-polynomial size) that she can specify. Examples of aggregate queries can be the sum/product of all function values belonging to an exponential-sized interval, or more generally, the sum/product of all function values on points for which some polynomial time predicate holds. Since the sets over which our function values are aggregated are super-polynomial in size, they cannot be directly computed by simply querying the function on individual points. AGG-PRFs cast in the framework of [GGN10] are (truthful, pseudo) implementations of random functions supporting aggregates as their “complex queries.” Indeed, our first example of an AGG-PRF for computing parities over exponential-sized intervals follows directly from [GGN10] under the assumption that

²Constrained PRFs are also known as Functional PRFs and as Delegatable PRFs.

one-way functions exist.

We show AGG-PRFs under various cryptographic hardness assumptions (one-way functions and DDH) for a number of types of aggregation operators such as sums and products and for a number of set systems including intervals, hypercubes, and (the supports of) restricted computational models such as decision trees and read-once Boolean formulas. We also show negative results: there are no AGG-PRFs for more expressive set systems such as (the supports of) CNF formulas. For a detailed description of our results, see Section 1.1.

In the second part of this paper, we embark on a study of the connection between the new augmented PRF constructions of recent years (constrained, related-key, aggregate) and the theory of computational learning. We recall at the outset that the fields of cryptography and machine learning share a curious historical relationship. The goals are in complete opposition and at the same time the aesthetics of the models, definitions and techniques bear a striking similarity. For example, a cryptanalyst can attack a cryptosystem using a range of powers from only seeing ciphertext examples to requesting to see decryptions of ciphertexts of her choice. Analogously, machine learning allows different powers to the learner such as random examples versus membership queries and shows that certain powers allow learners to learn concepts in polynomial time whereas others will fail. Even more directly, problems which pose challenges for machine learning such as Learning Parity with Noise (LPN) have been used as the underpinning for building secure cryptosystems, and as mentioned above [Val84] observes that the existence of PRFs in a complexity class \mathcal{C} implies the existence of concept classes in \mathcal{C} which can not be learned under membership queries, and [KV94] extends this direction to some public key constructions.

In the decades since the introduction of PAC learning, new computational learning models have been proposed, such as the recent “restriction access” model [DRWY12] which allows the learner to interact with the target concept by asking membership queries, but also to obtain an entire circuit that computes the concept on a random subset of the inputs. For example, in one shot, the learner can obtain a circuit that computes the concept class on all n -bit inputs that start with $n/2$ zeros. At the same time, the cryptographic research landscape has been swiftly moving in the direction of augmenting traditional PRFs and other cryptographic primitives to include higher functionalities. This brings to mind natural questions:

- *Can one leverage augmented pseudo-random function constructions to establish limits on what can and cannot be learned in augmented machine learning models?*
- *Going even further afield, can augmented cryptographic constructs suggest interesting learning models?*

We address these questions in the second part of this paper. For a detailed description of our findings, see Section 1.2.

1.1 Our Results: Aggregate Pseudo Random Functions

Aggregate Pseudo Random Functions (AGG-PRF) are indexed families of pseudo-random functions for which a distinguisher (who runs in time polynomial in the security parameter) can request and receive the value of an aggregate (for example, the sum or the product) of the function values over certain large sets and yet cannot distinguish oracle access to the function from oracle access to a truly random function. At the same time, given the function index (in other words, the secret key), one can compute such aggregates over potentially super-polynomial size sets in polynomial time.

Such an efficient aggregation algorithm cannot possibly exist for random functions. Thus, this is a PRF family that is very unlike random functions (in the sense of being able to efficiently aggregate over superpolynomial size sets), and yet is computationally indistinguishable from random functions.

To make this notion precise, we need two ingredients. Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda>0}$ where each $\mathcal{F}_\lambda = \{f_K : \mathcal{D}_\lambda \rightarrow \mathcal{R}_\lambda\}_{K \in \mathcal{K}_\lambda}$ is a collection of functions on a domain \mathcal{D}_λ to a range \mathcal{R}_λ , computable in time $\text{poly}(\lambda)$.³ The first ingredient is a collection of sets (also called a set system) $\mathcal{S} = \{S \subseteq \mathcal{D}\}$ over which the aggregates can be efficiently computed given the index K of the function. The second ingredient is an aggregation function $\Gamma : \mathcal{R}^* \rightarrow \{0,1\}^*$ which takes as input a tuple of function values $\{f(x) : x \in S\}$ for some set $S \in \mathcal{S}$ and outputs the aggregate $\Gamma(f(x_1), \dots, f(x_{|S|}))$.

The sets are typically super-polynomially large, but are efficiently recognizable. That is, for each set S , there is a corresponding $\text{poly}(\lambda)$ -size circuit C_S that takes as input an $x \in \mathcal{D}$ and outputs 1 if and only if $x \in S$.⁴ Throughout this paper, we will consider relatively simple aggregate functions, namely we will treat the range of the functions as an Abelian group, and will let Γ denote the group operation on its inputs. Note that the input to Γ is super-polynomially large (in the security parameter λ), making the aggregate computation non-trivial.

This family of functions, equipped with a set system \mathcal{S} and an aggregation function Γ is called an aggregate PRF family (AGG-PRF) if the following two requirements hold:

1. *Aggregatability*: There exists a polynomial (in the security parameter λ) time algorithm that given an index K to the PRF $f_K \in \mathcal{F}$ and a circuit C_S that recognizes a set $S \in \mathcal{S}$, can compute Γ over the PRF values $f_K(x)$ for all $x \in S$. That is, it can compute

$$\text{AGG}_{K,\Gamma}(S) := \Gamma_{x \in S} f_K(x)$$

2. *Pseudorandomness*: No polynomial-time distinguisher which can specify a set $S \in \mathcal{S}$ as a query and can receive as an answer either $\text{AGG}_{K,\Gamma}(S)$ for a random function $f_K \in \mathcal{F}$ or $\text{AGG}_{h,\Gamma}(S)$ for a truly random functions h , can distinguish between the two cases.

We show a number of constructions of AGG-PRF for various set systems under different cryptographic assumptions. We describe our constructions below, starting from the least expressive set system.

Interval Sets. We first present AGG-PRFs over interval set systems with respect to aggregation functions that compute any group operation. The construction can be based on any (standard) PRF family.

Theorem 1.1 (Group summation over intervals, from one-way functions [GGN10]).⁵ *Assume one-way functions exist. Then, there exists an AGG-PRF family that maps \mathbb{Z}_p to a group G , with respect to a collection of sets defined by intervals $[a, b] \subseteq \mathbb{Z}_p$ and the aggregation function computing the group operation on G .*

³In this informal exposition, for the sake of brevity, we will sometimes omit the security parameter and refrain from referring to ensembles.

⁴All the sets we consider are efficiently recognizable, and we use the corresponding circuit as the representation of the set. We occasionally abuse notation and use S and C_S interchangeably.

⁵Observed even earlier by Reingold and Naor and appeared in [GGI⁺02] in the context of small space streaming algorithms

The construction works as follows. Let $F : \{0, 1\}^n \times \{0, 1\}^n \rightarrow \{0, 1\}$ be a (standard) pseudo-random function family based on the existence of one-way functions [GGM86, HILL99]. Construct an AGG-PRF family G supporting efficient computation of group aggregation functions. Define

$$G(k, x) = F(k, x) - F(k, x - 1)$$

To aggregate G , set

$$\sum_{x \in [a, b]} G(k, x) = F(k, b) - F(k, a - 1)$$

Given k , this can be efficiently evaluated.

Another construction from [GGN10] achieves summation over the integers for PRFs whose range is $\{0, 1\}$. We omit the details of the construction, but state the theorem for completeness.

Theorem 1.2 (Integer summation over intervals, from one-way functions [GGN10]). *Assume one-way functions exist. Then, there exists an AGG-PRF family that maps \mathbb{Z}_{2^λ} to $\{0, 1\}$, with respect to a collection of sets defined by intervals $[a, b] \subseteq \mathbb{Z}_{2^\lambda}$ and the aggregation function computing the summation over \mathbb{Z} .*

Hypercubes. We next construct AGG-PRFs over hypercube set systems. This partially addresses Open Problem 5.4 posed in [GGN10], whether one can efficiently implement a random function with range $\{0, 1\}$ with complex queries that compute parities over the function values on hypercubes. Under subexponential DDH hardness, Theorem 1.3 answers the question for products rather than parities for a function whose range is a DDH group.

Throughout this section, we take $\mathcal{D}_\lambda = \{0, 1\}^\ell$ for some polynomial $\ell = \ell(\lambda)$. A hypercube $S_{\mathbf{y}}$ is defined by a vector $\mathbf{y} \in \{0, 1, \star\}^\ell$ as

$$S_{\mathbf{y}} = \{\mathbf{x} \in \{0, 1\}^\ell : \forall i, y_i = \star \text{ or } x_i = y_i\}$$

We present a construction under the sub-exponential DDH assumption.

Theorem 1.3 (Hypercubes from DDH). *Let $\mathcal{HC} = \{\mathcal{HC}_{\ell(\lambda)}\}_{\lambda > 0}$ where $\mathcal{HC}_\ell = \{0, 1, \star\}^\ell$ be the set of hypercubes on $\{0, 1\}^\ell$. Then, there is a construction of AGG-PRF supporting the set system \mathcal{HC} with the product aggregation function, assuming the subexponential DDH assumption.*

We sketch the construction from DDH below. Our DDH construction is the Naor-Reingold PRF [NR04]. Namely, the function is parametrized by an ℓ -tuple $\vec{k} = (k_1, \dots, k_\ell)$ and is defined as

$$F(\vec{k}, x) = g^{\prod_{i: x_i=1} k_i}$$

Let us illustrate aggregation over the hypercube $\mathbf{y} = (1, 0, \star, \star, \dots, \star)$. To aggregate the function F , observe that

$$\begin{aligned} \prod_{\{x: x_1=1, x_2=0\}} F(\vec{k}, x) &= \prod_{\{x: x_1=1, x_2=0\}} g^{\prod_{i: x_i=1} k_i} \\ &= g^{\sum_{\{x: x_1=1, x_2=0\}} \prod_{i: x_i=1} k_i} \\ &= g^{(k_1)(1)(k_2+1)(k_3+1)\dots(k_\ell+1)} \end{aligned}$$

which can be efficiently computed given \vec{k} .

Decision Trees. A decision tree T on ℓ variables is a binary tree where each internal node is labeled by a variable x_i , the leaves are labeled by either 0 or 1, one of the two outgoing edges of an internal node is labeled 0, and the other is labeled 1. Computation of a decision tree on an input (x_1, \dots, x_ℓ) starts from the root, and at each internal node n , proceeds by taking either the 0-outgoing edge or 1-outgoing edge depending on whether $x_n = 0$ or $x_n = 1$, respectively. Finally, the output of the computation is the label of the leaf reached through this process. The size of a decision tree is the number of nodes in the tree.

A decision tree T defines a set $S = S_T = \{x \in \{0, 1\}^\ell : T(x) = 1\}$. We show how to compute product aggregates over sets defined by polynomial size decision trees, under the subexponential DDH assumption.

The construction is simply a result of the observation that the set $S = S_T$ can be written as a disjoint union of polynomially many hypercubes. Computing aggregates over each hypercube and multiplying the results together gives us the decision tree aggregate.

Theorem 1.4 (Decision Trees from DDH). *Assuming the sub-exponential hardness of the decisional Diffie-Hellman assumption, there is an AGG-PRF that supports aggregation over sets recognized by polynomial-size decision trees.*

Read-Once Boolean Formulas. Finally, we show a construction of AGG-PRF over read-once Boolean formulas, the most expressive of our set systems, under the subexponential DDH assumption. A read-once Boolean formula is a Boolean circuit composed of AND, OR and NOT gates with fan-out 1, namely each input literal feeds into at most one gate, and each gate output feeds into at most one other gate. Thus, a read-once formula can be written as a binary tree where each internal node is labeled with an AND or OR gate, and each literal (variable or its negation) appears in at most one leaf.

Theorem 1.5 (Read-Once Boolean Formulas from DDH). *Under the subexponential decisional Diffie-Hellman assumption, there is an AGG-PRF that supports aggregation over sets recognized by read-once Boolean formulas.*

Our aggregate PRF is, once again, the Naor-Reingold PRF. The index of the PRF consists of a $(\ell + 1)$ -tuple of integers in \mathbb{Z}_p , namely $\vec{K} = (K_0, \dots, K_\ell) \in \mathbb{Z}_p^{\ell+1}$. The function is defined as

$$f_{\vec{K}}(x) = g^{K_0 \prod_{i \in [\ell]} K_i^{x_i}}$$

We compute aggregates by recursion on the levels of the formula. We start by noting that it is enough to compute

$$A(C, 1) := \sum_{x: C(x)=1} \prod_{i \in [1 \dots \ell]} K_i^{x_i}$$

because once this is done, it is easy to compute

$$\prod_{x: C(x)=1} f_{\vec{K}}(x) = g^{K_0 \cdot A(C, 1)}$$

For the purposes of this informal exposition, assume that ℓ is a power of two. Let C be the formula, with either $C = C_L \wedge C_R$ or $C = C_L \vee C_R$ for subformula C_L and C_R . We show how to recursively compute $A(C, 1)$ for these sub-circuits and thus for C .

Limits of Aggregation. A natural question to ask is whether one can support aggregation over sets defined by general circuits. It is however easy to see that you cannot support any class of circuits for which deciding satisfiability is hard (for example, AC^0), or even ones for which counting the number of SAT assignments is hard (DNFs, for example) as follows. Suppose C is a circuit which is either unsatisfiable or has a unique SAT assignment. Solving satisfiability for such circuits is known to be sufficient to solve SAT in general [VV86]. The algorithm for SAT simply runs the aggregator with a random PRF key K , and outputs YES if and only if the aggregator returns a non-zero value. Note that if the formula is unsatisfiable, we will always get 0 from the aggregator. Otherwise, we get $f_k(x)$, where x is the (unique) satisfying assignment. Now, this might end up being 0 accidentally, but cannot be 0 always since otherwise, we will turn it into a PRF distinguisher. The distinguisher has the satisfying assignment hardcoded into it non-uniformly, and it simply checks if $PRF_K(x)$ is 0.

Theorem 1.6 (Impossibility for General Set Systems). *Suppose there is an efficient algorithm which on an index for $f \in \mathcal{F}$, a set system defined by $\{x : C(x) = 1\}$ for a polynomial size Boolean circuit C , and an aggregation function Γ , outputs the $\Gamma_{x:C(x)=1}f(x)$. Then, there is efficient algorithm that takes circuits C as input and w.h.p. over its coins, decides satisfiability for C .*

1.1.1 Related Work to Aggregate PRFs

As described above, the work of [GGN10] studies the general question of how one can efficiently construct random, “close-to” random, and “pseudo-random” large objects, such as functions or graphs, which “truthfully” obey global combinatorial properties rather simply appearing to do so to a polynomial time observer.

Formally, using the [GGN10] terminology, a PRF is a *pseudo-implementation* of a random function, and an AGG-PRF is a pseudo-implementation of a “random function that also answers aggregate queries” (as we defined them). Furthermore, the aggregatability property of AGG-PRF implies it is a *truthful* pseudo-implementation of such a function. Whereas in this work, we restrict our attention to aggregate queries, [GGN10] considers additional “complex-queries,” such as in the case of a uniformly selected N node graph, providing a clique of size $\log_2 N$ that contains the queried vertex in addition to answering adjacency queries.

Our notion of aggregate PRFs bears resemblance to the notion of “algebraic PRFs” defined in the work of Benabbas, Gennaro and Vahlis [BGV11]. There are two main differences. First, algebraic PRFs support efficient aggregation over very specific subsets, whereas our constructions of aggregate PRFs support expressive subset classes, such as subsets recognized by hypercubes, decision trees and read-once Boolean formulas. Secondly, in the security notion for aggregate PRFs, the adversary obtains access to an oracle that computes the function as well as one that computes the aggregate values over super-polynomial size sets, whereas in algebraic PRFs, the adversary is restricted to accessing the function oracle alone. Our constructions from DDH use an algebraic property of the Naor-Reingold PRF in a similar manner as in [BGV11].

1.2 Our Results: Augmented PRFs and Computational Learning

As discussed above, connections between PRFs and learning theory date back to the 80’s in the pioneering work of [Val84] showing that PRF in a complexity class C implies the existence of concept classes in C which can not be learned with membership queries. In the second part of this work, we study the implications of the slew of augmented PRF constructions of recent

years [BW13a, BGI14a, KPTZ13b, BC10, ABPP14] and our new aggregate PRF to computational learning.

1.2.1 Constrained PRFs and limits on Restriction Access learnability

Recently, Dvir, Rao, Wigderson, and Yehudayoff [DRWY12] introduced a new learning model where the learner is allowed non-black-box information on the computational device (such as circuits, DNF, formulas) that decides the concept; their learner receives a simplified device resulting from partial assignments to input variables (i.e. restrictions). These partial restrictions lie somewhere in between function evaluation (full restrictions) which correspond to learning with membership queries and the full description of the original device (the empty restriction). The work of [DRWY12] studies a PAC version of restriction access, called PAC_{RA} , where the learner receives the circuit restricted with respect to random partial assignments. They show that both decision trees and DNF formulas can be learned efficiently in this model. Indeed, the PAC_{RA} model seems like quite a powerful generalization, if not too unrealistic, of the traditional PAC learning model, as it returns to the learner a computational description of the simplified concept.

Yet, in this section we will show limitations of this computational model under cryptographic assumptions. We show that the *constrained pseudo-random function families* introduced recently in [BW13b, BGI14b, KPTZ13a] naturally define a concept class which is not learnable by an even stronger variant of the restriction access learning model which we define. In the stronger variant, which we name *membership queries with restriction access* (MQ_{RA}) the learner can adaptively specify any restriction of the circuit from a specified class of restrictions \mathcal{S} and receive the simplified device computing the concept on this restricted domain in return. As this setting requires substantial notation, we define this new model very informally, and defer the formal definitions and theorems to the full version.

Definition 1.1 (Membership queries with restriction access (MQ_{RA})). *Let $\mathcal{C} : X \rightarrow \{0,1\}$ be a concept class, and $\mathcal{S} = \{S \subseteq X\}$ be a collection of subsets of the domain. \mathcal{S} is the set of allowable restrictions for concepts $f \in \mathcal{C}$. Let Simp be “simplification rule” which, for a concept f and restriction S outputs a “simplification” of f restricted to S .*

An algorithm \mathcal{A} is an $(\epsilon, \delta, \alpha)$ - MQ_{RA} learning algorithm for representation class \mathcal{C} with respect to a restrictions in \mathcal{S} and simplification rule Simp if, for every $f \in \mathcal{C}$, $\Pr[\mathcal{A}^{\text{Simp}(f, \cdot)} = h] \geq 1 - \delta$ where h is an ϵ -approximation to f – and furthermore, \mathcal{A} only requests restrictions for an α -fraction of the whole domain X .

Informally, constrained PRFs are PRFs with two additional properties: 1) for any subset S of the domain in a specified collection \mathcal{S} , a *constrained key* K_S can be computed, knowledge of which enables efficient evaluation of the PRF on S ; and 2) even with knowledge of constrained keys K_{S_1}, \dots, K_{S_m} for the corresponding subsets, the function retains pseudo-randomness on all points not covered by any of these sets. Connecting this to restriction access, the constrained keys will allow for generation of restriction access examples (restricted implementations with fixed partial assignments) and the second property implies that those examples do not aid in the learning of the function.

Theorem 1.7 (Informal). *Suppose \mathcal{F} is a family of constrained PRFs which can be constrained to sets in \mathcal{S} . If \mathcal{F} is computable in circuit complexity class \mathcal{C} , then \mathcal{C} is hard to MQ_{RA} -learn with restrictions in \mathcal{S} .*

Corollary 1.8 (Informal). *Existing constructions of constrained PRFs [BW13a] yield the following corollaries:*

- *If one-way functions exist, then poly-sized circuits can not be learned with restrictions on sub-intervals of the input-domain; and*
- *Assuming the sub-exponential hardness of the multi-linear Diffie-Hellman problem, NC^1 can-not be learned with restriction on hypercubes.*

1.2.2 New Learning Models Inspired by the Study of PRFs

We proceed to define two new learning models inspired by recent directions in cryptography. The first model is the *related concept* model inspired by work into related-key attacks in cryptography. While we have cryptography and lower bounds in mind, we argue that this model is in some ways natural. The second model, learning with *aggregate queries*, is directly inspired by our development of aggregate pseudo-random functions in this work; rather than being a natural model in its own right, this model further illustrates how cryptography and learning are duals in many senses.

The Related Concept Learning Model The idea that some functions or concepts are related to one another is quite natural. For a DNF formula, for instance, related concepts may include formulas where a clause has been added or formulas where the roles of two variables are swapped. For a decision tree, we could consider removing some accepting leaves and examining the resulting behavior. For a circuit, a related circuit might alter internal gates or fix the values on some wires. A similar phenomena occurs in cryptography, where secret keys corresponding to different instances of the same cryptographic primitive or even secret keys of different cryptographic primitives are related (if, for example, they were generated by a pseudo random process on the same seed).

We propose a new computational learning model where the learner is explicitly allowed to specify membership queries not only for the concept to be learned, but also for “related” concepts, given by a class of allowed transformations on the concept. We will show both a separation from membership queries, and a general negative result in the new model. Based on recent constructions of related-key secure PRFs by Bellare and Cash [BC10] and Abdalla et al [ABPP14], we demonstrate concept classes for which access to these related concepts is of no help.

To formalize the related concept learning model, we will consider *keyed concept classes* – classes indexed by a set of keys. This will enable the study of related concepts by instead considering concepts whose keys are related in some way. Most generally, we think of a key as a succinct representation of the computational device which decides the concept. This is a general framework; for example, we may consider the bit representation of a particular log-depth circuit as a key for a concept in the concept class NC^1 . For a concept f_k in concept class \mathcal{C} , we allow the learner to query a membership oracle for f_k and also for ‘related’ concepts $f_{\phi(k)} \in \mathcal{C}_K$ for ϕ in a specified class of allowable functions Φ . For example: let $K = \{0, 1\}^\lambda$ and let $\Phi^\oplus = \{\phi_\Delta : k \mapsto k \oplus \Delta\}_{\Delta \in \{0, 1\}^\lambda}$. Informally:

Definition 1.2 (Φ -Related-Concept Learning Model (Φ -RC)). *For \mathcal{C}_K a keyed concept class, let $\Phi = \{\phi : K \rightarrow K\}$ be a set of functions on K that contains the identity function id . A related-concept oracle RC_k , on query (ϕ, x) , responds with $f_{\phi(k)}(x)$, for all $\phi \in \Phi$ and $x \in X$.*

An algorithm A is an (ϵ, δ) - Φ -RK learning algorithm for a \mathcal{C}_k if, for every $k \in K$, when given access to the oracle $RK_k(\cdot)$, the algorithm A outputs with probability at least $1 - \delta$ a function $h : \{0, 1\}^n \rightarrow \{0, 1\}$ that ϵ -approximates f_k .

Yet again, we are able to demonstrate the limitations of this model using the power of a strong type of pseudo-random function. We show that *related-key secure PRF families* (RKA-PRF) defined and instantiated in [BC10] and [ABPP14] give a natural concept class which is not learnable with related key queries. RKA-PRFs are defined with respect to a set Φ of functions on the set of PRF keys. Informally, the security notion guarantees that for a randomly selected key k , no efficient adversary can distinguish oracle access to f_k and $f_{\phi(k)}$ (for many adaptively chosen functions $\phi \in \Phi$) from an oracle that returns completely random values. We leverage this strong pseudo-randomness property to show hard-to-learn concepts in the related concept model.

Theorem 1.9 (Informal). *Suppose \mathcal{F} is a family of RKA-PRFs with respect to related-key functions Φ . If \mathcal{F} is computable in circuit complexity class \mathcal{C} , then \mathcal{C} is hard to learn in the Φ' -RC model for some Φ' .*

Existing constructions of RKA-PRFs [ABPP14] yield the following corollary:

Corollary 1.10 (Informal). *Assuming the hardness of the DDH problem, and collision-resistant hash functions, NC^1 is hard to Φ -RC-learn for an class of affine functions Φ .*

The Aggregate Learning Model The other learning model we propose is inspired by our aggregate PRFs. Here, we consider a new extension to the power of the learning algorithm. Whereas membership queries are of the form “What is the label of an example x ?”, we grant the learner the power to request the evaluation of simple functions on tuples of examples (x_1, \dots, x_n) such as “How many of x_1, \dots, x_n are in \mathcal{C} ?” or “Compute the product of the labels of x_1, \dots, x_n ?”. Clearly, if n is polynomial then this will result only a polynomial gain in the query complexity of a learning algorithm in the best case. Instead, we propose to study cases when n may be super-polynomial, but the description of the tuples is succinct. For example, the learning algorithm might query the number of x ’s in a large interval that are positive examples in the concept.

As with the restriction access and related concept models – and the aggregate PRFs we define in this work – the Aggregate Queries (AQ) learning model will be considered with restrictions to both the types of aggregate functions Γ the learner can query, and the sets \mathcal{S} over which the learner may request these functions to be evaluated on. We now present the AQ learning model informally:

Definition 1.3 ((Γ, \mathcal{S}) -Aggregate Queries (AQ) Learning). *Let $\mathcal{C} : X \rightarrow \{0, 1\}$ be a concept class, and let \mathcal{S} be a collection of subsets of X . Let $\Gamma : \{0, 1\}^* \rightarrow V$ be an aggregation function. For $f \in \mathcal{C}$, let AGG_f be an “aggregation” oracle, which for $S \in \mathcal{S}$, returns $\Gamma_{x \in S} f(x)$. Let MEM_f be the membership oracle, which for input x returns $f(x)$.*

An algorithm \mathcal{A} is an (ϵ, δ) - (Γ, \mathcal{S}) -AQ learning algorithm for \mathcal{C} if for every $f \in \mathcal{C}$,

$$\Pr[\mathcal{A}^{\text{MEM}_f(\cdot), \text{AGG}_f(\cdot)} = h] \geq 1 - \delta$$

where h is an ϵ -approximation to f .

Initially, AQ learning is reminiscent of learning with statistical queries (SQ). In fact, this apparent connection inspired this portion of our work. But the AQ setting is in fact incomparable to

SQ learning, or even the weaker “statistical queries that are independent of the target” as defined in [BF02]. On the one hand, AQ queries provide a sort of noiseless variant of SQ, giving more power to the AQ learner; on the other hand, the AQ learner is restricted to aggregating over sets in \mathcal{S} , whereas the SQ learner is not restricted in this way, thereby limiting the power of the AQ learner. The AQ setting where \mathcal{S} contains every subset of the domain is indeed a noiseless version of “statistical queries independent of the target,” but even this model is a restricted version of SQ. This does raise the natural question of a noiseless version of SQ and its variants; hardness results in such models would be interesting in that they would suggest that the hardness comes not from the noise but from an inherent loss of information in statistics/aggregates.

We will show both a simple separation from learning with membership queries (in the full version), and under cryptographic assumptions, a general lower bound on the power of learning with aggregate queries. The negative examples will use the results in Section 1.1.

Theorem 1.11. *Let \mathcal{F} be a boolean-valued aggregate PRF with respect to set system \mathcal{S} and aggregation function Γ . If \mathcal{F} is computable in complexity class \mathcal{C} , then \mathcal{C} is hard to (Γ, \mathcal{S}) -AQ learn.*

Corollary 1.12. *Using the results from Section 3, we get the following corollaries:*

- *The existence of one way functions implies that P/poly is hard to $(\sum, \mathcal{S}_{[a,b]})$ -AQ learn, with $\mathcal{S}_{[a,b]}$ the set of sub-intervals of the domain as defined in Section 3.*
- *The DDH assumption implies that NC^1 is hard to $(\sum, \mathcal{S}_{[a,b]})$ -AQ learn, with $\mathcal{S}_{[a,b]}$ being the set of sub-intervals of the domain as defined in Section 3.*
- *The subexponential DDH Assumption implies that NC^1 is hard to (\prod, \mathcal{R}) -AQ learn, with \mathcal{R} the set of read-once boolean formulas defined in Section 3.*

Open Questions. As discussed in the introduction, augmented pseudo-random functions often have powerful and surprising applications, perhaps the most recent example being constrained PRFs [BW13a, KPTZ13a, BGI14a]. Perhaps the most obvious open question that emerges from this work is to find applications for aggregate PRFs. We remark that a primitive similar to aggregate PRFs was used in [BGV11] to construct delegation protocols.

Perhaps a more immediate concern is that all our aggregate PRF constructions (except for intervals) requires sub-exponential hardness assumptions. We view it as an important open question to base these constructions on polynomial assumptions.

In this work we restricted our attention to particular types of aggregation functions and subsets over which the aggregation takes place, although our definition captures more general scenarios. We looked at aggregation functions that compute group operations over Abelian groups. Can we support more general aggregation functions that are not restricted to group operations, for example the majority aggregation function, or even non-symmetric aggregation functions? We show positive results for intervals, hypercubes, and sets recognized by read-once formulas and decision trees. On the other hand, we show that it is unlikely that we can support general sets, for example sets recognized by CNF formulas. This almost closes the gap between what is possible and what is hard. A concrete open question in this direction is to construct an aggregate PRF computing summation over an Abelian group for sets recognized by DNFs, or provide evidence that this cannot be done.

Organization. This paper is organized into two parts that can be read essentially independently of each other. In the first part (Sections 2 and 3), we present the definition and constructions of aggregate pseudo-random functions. In the second part (Section 4), we show connections between various notions of augmented PRFs and their applications to augmented learning models.

2 Aggregate PRF

We will let λ denote the security parameter throughout this paper.

Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda>0}$ be a function family where each function $f \in \mathcal{F}_\lambda$ maps a domain \mathcal{D}_λ to a range \mathcal{R}_λ . An *aggregate function* family is associated with two objects:

1. an ensemble of sets $\mathcal{S} = \{\mathcal{S}_\lambda\}_{\lambda>0}$ where each \mathcal{S}_λ is a collection of subsets of the domain $S \subseteq \mathcal{D}_\lambda$; and
2. an “aggregation function” $\Gamma_\lambda : (\mathcal{R}_\lambda)^* \rightarrow \mathcal{V}_\lambda$ that takes a tuple of values from the range \mathcal{R}_λ of the function family and “aggregates” them to produce a value in an output set \mathcal{V}_λ .

Let us now make this notion formal. To do so, we will impose restrictions on the set ensembles and the aggregation function. First, we require set ensemble \mathcal{S}_λ to be *efficiently recognizable*. That is, there is a polynomial-size Boolean circuit family $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda>0}$ such that for any set $S \in \mathcal{S}_\lambda$ there is a circuit $C = C_S \in \mathcal{C}_\lambda$ such that $x \in S$ if and only if $C(x) = 1$. Second, we require our aggregation functions Γ to be efficient in the length of its inputs, and symmetric; namely the output of the function does not depend on the order in which the inputs are fed into it. Summation over an Abelian group is an example of a possible aggregation function. Third and finally, elements in our sets \mathcal{D}_λ , \mathcal{R}_λ , and \mathcal{V}_λ are all representable in $\text{poly}(\lambda)$ bits, and the functions $f \in \mathcal{F}_\lambda$ are computable in $\text{poly}(\lambda)$ time.

Define the aggregate function $\text{AGG} = \text{AGG}_{f, \mathcal{S}_\lambda, \Gamma_\lambda}^\lambda$ that is indexed by a function $f \in \mathcal{F}_\lambda$, takes as input a set $S \in \mathcal{S}_\lambda$ and “aggregates” the values of $f(x)$ for all $x \in S$. That is, $\text{AGG}(S)$ outputs

$$\Gamma(f(x_1), f(x_2), \dots, f(x_{|S|}))$$

where $S = \{x_1, \dots, x_{|S|}\}$. More precisely, we have

$$\begin{aligned} \text{AGG}_{f, \mathcal{S}_\lambda, \Gamma_\lambda}^\lambda : \mathcal{S}_\lambda &\rightarrow \mathcal{V}_\lambda \\ S &\mapsto \Gamma_{x_i \in S}(f(x_1), \dots, f(x_{|S|})) \end{aligned}$$

We will furthermore require that the AGG can be computed in $\text{poly}(\lambda)$ time. We require this in spite of the fact that the sets over which the aggregation is done can be exponentially large! Clearly, such a thing is impossible for a random function f but yet, we will show how to construct *pseudo-random* function families that support efficient aggregate evaluation. We will call such a pseudo-random function (PRF) family an *aggregate PRF* family. In other words, our objective is two fold:

1. Allow anyone who knows the (polynomial size) function description to efficiently compute the aggregate function values over exponentially large sets; but at the same time,
2. Ensure that the function family is indistinguishable from a truly random function, even given an oracle that computes aggregate values.

A simple example of aggregates is that of computing the summation of function values over sub-intervals of the domain. That is, let domain and range be \mathbb{Z}_p for some $p = p(\lambda)$, let the family of subsets be $\mathcal{S}_\lambda = \{[a, b] \subseteq \mathbb{Z}_p : a, b \in \mathbb{Z}_p; a \leq b\}$, and the aggregation function be $\Gamma_\lambda(y_1, \dots, y_k) = \sum_{i=1}^k y_i \pmod{p}$. In this case, we are interested in computing

$$\text{AGG}_{f, \mathcal{S}_\lambda, \text{sum}}^\lambda([a, b]) = \sum_{a \leq x \leq b} f(x)$$

We will, in due course, show both constructions and impossibility results for aggregate PRFs, but first let us start with the formal definition.

Definition 2.1 (Aggregate PRF). *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda>0}$ be a function family where each function $f \in \mathcal{F}_\lambda$ maps a domain \mathcal{D}_λ to a range \mathcal{R}_λ , \mathcal{S} be an efficiently recognizable ensemble of sets $\{\mathcal{S}_\lambda\}_{\lambda>0}$, and $\Gamma_\lambda : (\mathcal{R}_\lambda)^* \rightarrow \mathcal{V}_\lambda$ be an aggregation function. We say that \mathcal{F} is an (\mathcal{S}, Γ) -aggregate pseudorandom function family (also denoted (\mathcal{S}, Γ) -AGG-PRF) if there exists an efficient algorithm $\text{Aggregate}_{k, \mathcal{S}, \Gamma}(S)$: On input a subset $S \in \mathcal{S}$ of the domain, outputs $v \in \mathcal{V}$, such that*

- **Efficient aggregation:** *For every $S \in \mathcal{S}$, $\text{Aggregate}_{k, \mathcal{S}, \Gamma}(S) = \text{AGG}_{k, \mathcal{S}, \Gamma}(S)$ where $\text{AGG}_{k, \mathcal{S}, \Gamma}(S) := \Gamma_{x \in S} F_k(x)$.⁶⁷*
- **Pseudorandomness:** *For all probabilistic polynomial-time (in security parameter λ) algorithms A , and for randomly selected key $k \in K$:*

$$\left| \Pr_{f \leftarrow \mathcal{F}_\lambda} [A^{f_k, \text{AGG}_{f_k, \mathcal{S}, \Gamma}}(1^\lambda)] - \Pr_{h \leftarrow \mathcal{H}_\lambda} [A^{h, \text{AGG}_{h, \mathcal{S}, \Gamma}}(1^\lambda)] \right| \leq \text{negl}(\lambda)$$

where \mathcal{H}_λ is the set of all functions $\mathcal{D}_\lambda \rightarrow \mathcal{R}_\lambda$.

Remark. In this work, we restrict our attention to aggregation functions that treat the range $\mathcal{V}_\lambda = \mathcal{R}_\lambda$ as an Abelian group and compute the group sum (or product) of its inputs. We denote this setting by $\Gamma = \sum$ (or \prod , respectively). Supporting other types of aggregation functions (ex: max, a hash) is a direction for future work.

2.1 A General Security Theorem for Aggregate PRFs

How does the security of a function family in the AGG-PRF game relate to security in the normal PRF game (in which A uses only the oracle f and not AGG_f)?

In this section, we show a general security theorem for aggregate pseudo-random functions. Namely, we show that any “sufficiently secure” PRF is also aggregation-secure (for any collection of efficiently recognizable sets and any group-aggregation operation), in the sense of Definition 2.1, by way of an *inefficient* reduction (with overhead polynomial in the size of the domain). In Section 3, we will use this to construct AGG-PRFs from a subexponential-time hardness assumption on the DDH problem. We also show that no such *general* reduction can be efficient, by demonstrating a PRF family that is not aggregation-secure. As a general security theorem cannot be shown without the use of complexity leveraging, this suggests a natural direction for future study: to devise constructions for similarly expressive aggregate PRFs from polynomial assumptions.

⁶We omit subscripts on AGG and Aggregate when clear from context.

⁷AGG is defined to be the correct aggregate value, while Aggregate is the algorithm by which we compute the value AGG. We make this distinction because while a random function cannot be efficiently aggregated, the aggregate value is still well-defined.

Lemma 2.1. *Let $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda>0}$ be a pseudo-random function family where each function $f \in \mathcal{F}_\lambda$ maps a domain \mathcal{D}_λ to a range \mathcal{R}_λ . Suppose there is an adversary A that runs in time $t_A = t_A(\lambda)$ and achieves an advantage of $\epsilon_A = \epsilon_A(\lambda)$ in the aggregate PRF security game for the family \mathcal{F} with an efficiently recognizable set system \mathcal{S}_λ and an aggregation function Γ_λ that is computable in time polynomial in its input length. Then, there is an adversary B that runs in time $t_B = t_A + \text{poly}(\lambda, |\mathcal{D}_\lambda|)$ and achieves an advantage of $\epsilon_B = \epsilon_A$ in the standard PRF game for the family \mathcal{F} .*

Proof. Let $f_K \leftarrow \mathcal{F}_\lambda$ be a random function from the family \mathcal{F}_λ . We construct the adversary B which is given access to an oracle \mathcal{O} which is either f_K or a uniformly random function $h : \mathcal{D}_\lambda \rightarrow \mathcal{R}_\lambda$.

B works as follows: It queries the PRF on all inputs $x \in \mathcal{D}_\lambda$, builds the function table T_K of f_K and runs the adversary A , responding to its queries as follows:

1. Respond to its PRF query $x \in \mathcal{D}_\lambda$ by returning $T_K[x]$; and
2. Respond to its aggregate query (Γ, S) by (a) going through the table to look up all x such that $x \in S$; and (b) applying the aggregation function honestly to these values.

Finally, when A halts and returns a bit b , B outputs the bit b and halts.

B takes $O(|\mathcal{D}_\lambda|)$ time to build the truth table of the oracle. For each aggregate query (Γ, S) , B first checks for each $x \in \mathcal{D}_\lambda$ whether $x \in S$. This takes $|\mathcal{D}_\lambda| \cdot \text{poly}(\lambda)$ time, since S is efficiently recognizable. It then computes the aggregation function Γ over $f(x)$ such that $x \in S$, taking $\text{poly}(|\mathcal{D}_\lambda|)$ time, since Γ is computable in time polynomial in its input length. The total time, therefore, is

$$t_B = t_A + \text{poly}(\lambda, |\mathcal{D}_\lambda|)$$

Clearly, when \mathcal{O} is the pseudo-random function f_K , B simulates an aggregatable PRF oracle to A , and when \mathcal{O} is a random function, B simulates an aggregate random oracle to A . Thus, B has the same advantage in the PRF game as A does in the aggregate PRF game. \square

The above gives an inefficient reduction from the PRF security of a function family \mathcal{F} to the AGG-PRF security of the same family running in time polynomial in the size of the domain. Can this reduction be made efficient; that is, can we replace $t_B = t_A + \text{poly}(\lambda)$ into the Lemma 2.1?

This is not possible. Such a reduction would imply that every PRF family that supports efficient aggregate functionality AGG is AGG-PRF secure; this is clearly false. Take for example a pseudorandom function family $\mathcal{F}_0 = \{f : \mathbb{Z}_{2p} \rightarrow \mathbb{Z}_p\}$ such that for all f , there is no x with $f(x) = 0$. It is possible to construct such a pseudorandom function family \mathcal{F}_0 (under the standard definition). While 0 is not in the image of any $f \in \mathcal{F}_0$, a random function with the same domain and range will, with high probability, have 0 in the image. For an aggregation oracle AGG_f computing *products* over \mathbb{Z}_p : $\text{AGG}_f(\mathbb{Z}_{2p}) \neq 0$ if $f \in \mathcal{F}_0$, while $\text{AGG}_f(\mathbb{Z}_{2p}) = 0$ with high probability for random f .

Thus, access to aggregates for products over \mathbb{Z}_p ⁸ would allow an adversary to trivially distinguish $f \in \mathcal{F}_0$ from a truly random map.

⁸Taken with respect to a set ensemble \mathcal{S} containing, as an element, the whole domain \mathbb{Z}_{2p} . While this is not necessary (a sufficiently large subset would suffice), it is the case for the ensembles \mathcal{S} we consider in this work.

2.2 Impossibility of Aggregate PRF for General Sets

It is natural to ask whether an aggregate PRF might be constructed for more general sets than we present in Section 3. There we constructed aggregate PRF for the sets of all satisfying assignments for read-once boolean formula and decision trees. As we show in the following, it is impossible to extend this to support the set of satisfying assignments for more general circuits.

Theorem 2.2. *Suppose there is an algorithm that has a PRF description K , a circuit C , and a fixed aggregation rule (sum over a finite field, say), and outputs the aggregate value*

$$\sum_{x:C(x)=1} f_K(x)$$

Then, there is an algorithm that takes circuits C as input and w.h.p. over its coins, decides the satisfiability of C .

Proof. The algorithm for SAT simply runs the aggregator with a randomly chosen K , and outputs YES if and only if the aggregator returns 1. The rationale is that if the formula is unsatisfiable, you will always get 0 from the aggregator.⁹ Otherwise, you will get $f_K(x)$, where x is the satisfying assignment. (More generally, $\sum_{x:C(x)=1} f_K(x)$). Now, this might end up being 0 accidentally, but cannot be 0 always since otherwise, you will get a PRF distinguisher. The distinguisher has the satisfying assignment hardcoded into it non-uniformly,¹⁰ and it simply checks if $f_K(x) = 0$. \square

This impossibility result can be generalized for efficient aggregation of functions that are not pseudo-random. For instance, if $f(x) \equiv 1$ was the constant function 1, the same computing the aggregate over f satisfying inputs to C would not only reveal the satisfiability of C , but even the number of satisfying assignments! In the PRF setting though, it seems that aggregates only reveal the (un)satisfiability of a circuit C , but not the number of satisfying assignments. Further studying the relationship between the (not necessarily pseudo-random) function f , the circuit representation of C , and the tractability of computing aggregates is an interesting direction. A negative result for a class for which satisfiability (or even counting assignments) is tractable would be very interesting.

3 Constructions of aggregate PRF

In this section, we show several constructions of aggregate PRFs. In Section 3.1, we show as a warm-up a generic construction of aggregate PRFs for intervals (where the aggregation is any group operation). This construction is black-box: given any PRF with the appropriate domain and range, we construct a related family of aggregate PRFs and with no loss in security. In Section 3.2, we show a construction of aggregate PRFs for products over bit-fixing sets (hypercubes), from a strong decisional Diffie-Hellman assumption. We then generalize the DDH construction: in Section 3.3, to the class of sets recognized by polynomial-size decision trees; and in Section 3.4, to sets recognized by read-once Boolean formulas. In these last three constructions, we make use of Lemma 2.1 to argue security.

⁹This proof may be extended to the case when the algorithm's output is not restricted to be 0 when the input circuit C is unsatisfiable, and even arbitrary outputs for sufficiently expressive classes of circuits.

¹⁰As pointed out by one reviewer, for sufficiently expressive classes of circuits C , this argument can be made uniform. Specifically, we use distinguish the challenge y from a pseudo-random generator from random by choosing $C := C_y$ that is satisfiable if and only if y is in the PRG image, and modify the remainder of the argument accordingly.

3.1 Generic Construction for Interval Sets

Our first construction is from [GGN10]¹¹. The construction is entirely black-box: from any appropriate PRF family \mathcal{G} , we construct a related AGG-PRF family \mathcal{F} . Unlike the proofs in the sequel, this reduction exactly preserves the security of the starting PRF.

Let $\mathcal{G}_\lambda = \{g_K : \mathbb{Z}_{n(\lambda)} \rightarrow R_\lambda\}_{K \in \mathcal{K}_\lambda}$ be a PRF family, with $R = R_\lambda$ being a group where the group operation is denoted by \oplus ¹². We construct an aggregatable PRF $\mathcal{F}_\lambda = \{f_K\}_{K \in \mathcal{K}_\lambda}$ for which we can efficiently compute summation of $f_K(x)$ for all x in an interval $[a, b]$, for any $a \leq b \in \mathbb{Z}_n$. Let $\mathcal{S}_{[a,b]} = \{[a, b] \subseteq \mathbb{Z}_n : a, b \in \mathbb{Z}_n; a \leq b\}$ be the set of all interval subsets of \mathbb{Z}_n , $[a, b] = \{x \in \mathbb{Z}_n : a \leq x \leq b\}$. Define $\mathcal{F} = \{f_K : \mathbb{Z}_n \rightarrow R\}_{K \in \mathcal{K}}$ as follows:

$$f_K(x) = \begin{cases} g_K(0) & : x = 0 \\ g_K(x) \ominus g_K(x-1) & : x \neq 0 \end{cases}$$

Lemma 3.1. *Assuming that \mathcal{G} is a pseudo-random function family, \mathcal{F} is a $(\mathcal{S}_{[a,b]}, \oplus)$ -aggregate pseudo-random function family.*

Proof. It follows immediately from the definition of f_K that one can compute the summation of $f_K(x)$ over any interval $[a, b]$. Indeed, rearranging the definition yields

$$\sum_{x \in [0, b]} f_K(x) = g_K(b) \quad \text{and} \quad \sum_{x \in [a, b]} f_K(x) = g_K(b) \oplus -g_K(a-1)$$

We reduce the pseudo-randomness of \mathcal{F} to that of \mathcal{G} . The key observation is that each query to the f_K oracle as well as the aggregation oracle for f_K can be answered using at most two black-box calls to the underlying function g_K . By assumption on \mathcal{G} , replacing the oracle for g_K with a uniformly random function $h : \mathbb{Z}_n \rightarrow R$ is computationally indistinguishable. Furthermore, the function f defined by replacing g by h , namely

$$f'(x) = \begin{cases} h(0) & : x = 0 \\ h(x) \ominus h(x-1) & : x \neq 0 \end{cases}$$

is a truly random function. Thus, the simulated oracle with g_K replaced by h implements a uniformly random function that supports aggregate queries. Security according to Definition 2.1 follows immediately. \square

Another construction from the same work achieves summation over the integers for PRFs whose range is $\{0, 1\}$. We omit the details of the construction, but state the theorem for completeness.

Theorem 3.2 (Integer summation over intervals, from one-way functions [GGN10]). *Assume one-way functions exist. Then, there exists an $(\mathcal{S}_{[a,b]}, \sum)$ -AGG-PRF family that maps \mathbb{Z}_{2^λ} to $\{0, 1\}$, where \sum denotes summation over \mathbb{Z} .*

¹¹See Example 3.1 and Footnote 18

¹²The only structure of \mathbb{Z}_n we use is the *total order*. Our construction directly applies to any finite, totally-ordered domain D by first mapping D to \mathbb{Z}_n , preserving order.

3.2 Bit-Fixing Aggregate PRF from DDH

We now construct an aggregate PRF computing products for bit-fixing sets. Informally, our PRF will have domain $\{0, 1\}^{\text{poly}(\lambda)}$, and support aggregation over sets like $\{x : x_1 = 0 \wedge x_2 = 1 \wedge x_7 = 0\}$. We will naturally represent such sets by a string in $\{0, 1, \star\}^{\text{poly}(\lambda)}$ with 0 and 1 indicating a fixed bit location, and \star indicating a free bit location. We call each such set a ‘hypercube.’ The PRF will have a multiplicative group \mathcal{G} as its range, and the aggregate functionality will compute group products.

Our PRF is exactly the Naor-Reingold PRF [NR04], for which we demonstrate efficient aggregation and security. We begin by stating the decisional Diffie-Hellman assumption.

Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda>0}$ be a family of groups of order $p = p(\lambda)$. The decisional Diffie-Hellman assumption for \mathcal{G} says that the following two ensembles are computationally indistinguishable:

$$\begin{aligned} & \{(\mathcal{G}_\lambda, g, g^a, g^b, g^{ab}) : G \leftarrow \mathcal{G}_\lambda; g \leftarrow G; a, b \leftarrow \mathbb{Z}_p\}_{\lambda>0} \\ & \approx_c \{(G, g, g^a, g^b, g^c) : G \leftarrow \mathcal{G}_\lambda; g \leftarrow G; a, b, c \leftarrow \mathbb{Z}_p\}_{\lambda>0} \end{aligned}$$

We say that the $(t(\lambda), \epsilon(\lambda))$ -DDH assumption holds if for every adversary running in time $t(\lambda)$, the advantage in distinguishing between the two distributions above is at most $\epsilon(\lambda)$.

3.2.1 Construction

Let $\mathcal{G} = \{\mathcal{G}_\lambda\}_{\lambda>0}$ be a family of groups of order $p = p(\lambda)$, each with a canonical generator g , for which the decisional Diffie Hellman (DDH) problem is hard. Let $\ell = \ell(\lambda)$ be a polynomial function. We will construct a PRF family $\mathcal{F}_\ell = \{\mathcal{F}_{\ell, \lambda}\}_{\lambda>0}$ where each function $f \in \mathcal{F}_{\ell, \lambda}$ maps $\{0, 1\}^{\ell(\lambda)}$ to \mathcal{G}_λ . Our PRF family is exactly the Naor-Reingold PRF [NR04]. Namely, each function f is parametrized by $\ell + 1$ numbers $\vec{K} := (K_0, K_1, \dots, K_\ell)$, where each $K_i \in \mathbb{Z}_p$.

$$f_{\vec{K}}(x_1, \dots, x_\ell) = g^{K_0 \prod_{i=1}^\ell K_i^{x_i}} = g^{K_0 \prod_{i: x_i=1} K_i} \in \mathcal{G}_\lambda$$

The aggregation algorithm **Aggregate** for bit-fixing functions gets as input the PRF key \vec{K} and a bit-fixing string $y \in \{0, 1, \star\}^\ell$ and does the following:

- Define the strings K'_i as follows:

$$K'_i = \begin{cases} 1 & \text{if } y_i = 0 \\ K_i & \text{if } y_i = 1 \\ 1 + K_i & \text{otherwise} \end{cases}$$

- Output $g^{K_0 \prod_{i=1}^\ell K'_i}$ as the answer to the aggregate query.

Letting $\mathcal{HC} = \{\mathcal{HC}_{\ell(\lambda)}\}_{\lambda>0}$ where $\mathcal{HC}_\ell = \{0, 1, \star\}^\ell$ is the set of hypercubes on $\{0, 1\}^\ell$, we now prove the following:

Theorem 3.3. *Let $\epsilon > 0$ be a constant, choose the security parameter $\lambda = \Omega(\ell^{1/\epsilon})$, and assume the $(2^{\lambda^\epsilon}, 2^{-\lambda^\epsilon})$ -hardness of DDH over the group \mathcal{G} . Then, the collection of functions \mathcal{F} defined above is a secure aggregate PRF with respect to the subsets \mathcal{HC} and the product aggregation function over \mathcal{G} .*

Correctness. We show that the answer we computed for an aggregate query $y \in \{0, 1, \star\}^\lambda$ is correct. Define the sets

$$\text{Match}(y) := \{x \in \{0, 1\}^\lambda : \forall i, y_i = \star \text{ or } x_i = y_i\} \text{ and } \text{Fixed}(y) := \{i \in [\lambda] : y_i \in \{0, 1\}\}$$

Thus, $\text{Match}(y)$ is the set of all 0-1 strings x that match all the fixed locations of y , but can take any value on the wildcard locations of y . $\text{Fixed}(y)$ is the set of all locations i where the bit y_i is fixed. Note that:

$$\begin{aligned} \text{AGG}(\vec{K}, y) &= \prod_{x \in \text{Match}(y)} f_{\vec{K}}(x) && \text{(by definition of AGG)} \\ &= \prod_{x \in \text{Match}(y)} g^{K_0 \prod_{i=1}^\ell K_i^{x_i}} && \text{(by definition of } f_{\vec{K}} \text{)} \\ &= g^{K_0 \sum_{x \in \text{Match}(y)} \prod_{i=1}^\ell K_i^{x_i}} \\ &= g^{K_0 \left(\prod_{i \in \text{Fixed}(y)} K_i^{y_i} \right) \cdot \left(\prod_{i \in [\ell] \setminus \text{Fixed}(y)} (1 + K_i) \right)} && \text{(inverting sums and products)} \\ &= g^{K_0 \prod_{i=1}^\ell K_i'} && \text{(by definition of } K_i') \\ &= \text{Aggregate}(\vec{K}, y) && \text{(by definition of Aggregate)} \end{aligned}$$

Security. We will rely on the following theorem from [NR04].

Theorem 3.4 (Theorem 4.1, [NR04]). *Suppose there is an adversary A that runs in time $t(\lambda)$ and has an advantage of $\gamma(\lambda)$ in the (regular) PRF game. Then, there is an adversary B that runs in time $\text{poly}(\lambda) \cdot t(\lambda)$ and breaks the DDH assumption with advantage $\gamma(\lambda)/\lambda$.*

The aggregate PRF security proof proceeds as follows. First, we choose the security parameter $\lambda = \Omega(\ell^{1/\epsilon})$ as in the theorem statement. We use Lemma 2.1 to conclude that if there is an adversary distinguisher D breaking the aggregate PRF security of \mathcal{F} in $\text{poly}(\lambda)$ time with $1/\text{poly}(\lambda)$ advantage, then there is an adversary A that breaks the regular PRF security of \mathcal{F} in $\text{poly}(\lambda) \cdot 2^{O(\ell)} = \text{poly}(\lambda) \cdot 2^{\lambda^\epsilon} = 2^{O(\lambda^\epsilon)}$ time with $1/\text{poly}(\lambda)$ advantage. Using Theorem 3.4 now tells us that there is an adversary B that wins the DDH distinguishing game in $2^{O(\lambda^\epsilon)}$ time with $1/\text{poly}(\lambda)$ advantage, breaking the subexponential DDH assumption. This establishes the aggregate security of the PRF and thus Theorem 3.3.

Obtaining a security proof based on polynomial assumptions is an interesting open question.

3.3 Decision Trees

We generalize the previous construction from DDH to support sets specified by polynomial-sized decision trees by observing that such decision trees can be written as disjoint unions of hypercubes.

A decision tree family \mathcal{T}_λ of size $p(\lambda)$ over $\ell(\lambda)$ variables consists of binary trees with at most $p(\lambda)$ nodes, where each internal node is labeled with a variable x_i for $i \in [\ell]$, the two outgoing edges of an internal node are labeled 0 and 1, and the leaves are labeled with 0 or 1. On input an $x \in \{0, 1\}^\ell$, the computation of the decision tree starts from the root, and upon reaching an internal node n labeled by a variable x_i , takes either the 0-outgoing edge or the 1-outgoing edge out of the node n , depending on whether x_i is 0 or 1, respectively.

We now show how to construct a PRF family $\mathcal{F}_\ell = \{\mathcal{F}_{\ell, \lambda}\}_{\lambda > 0}$ where each $\mathcal{F}_{\ell, \lambda}$ consists of functions that map $\mathcal{D}_\lambda := \{0, 1\}^\ell$ to a group \mathcal{G}_λ , that supports aggregation over sets recognized by decision trees. That is, let $\mathcal{S}_\lambda = \{S \subseteq \{0, 1\}^\ell : \exists \text{ a decision tree } T_S \in \mathcal{T}_\lambda \text{ that recognizes } S\}$.

Our construction uses a hypercube-aggregate PRF family \mathcal{F}'_ℓ as a sub-routine. First, we need the following simple lemma.

Lemma 3.5 (Decision Trees as Disjoint Unions of Hypercubes). *Let $S \subseteq \{0, 1\}^\ell$ be recognized by a decision tree T_S of size $p = p(\lambda)$. Then, S is a disjoint union of at most p hypercubes H_{y_1}, \dots, H_{y_p} , where each $y_i \in \{0, 1, \star\}^\ell$ and $H_{y_i} = \text{Match}(y_i)$. Furthermore, given T_S , one can in polynomial time compute these hypercubes.*

Given the lemma, **Aggregate** is simple: on input a set S represented by a decision tree T_S , compute the disjoint hypercubes H_{y_1}, \dots, H_{y_p} . Run the hypercube aggregation algorithm to compute

$$g_i \leftarrow \text{Aggregate}_{\mathcal{F}}(K, y_i)$$

and outputs $g := \prod_{i=1}^p g_i$.

Basing the construction on the hypercube-aggregate PRF scheme from Section 3.2, we get a decision tree-aggregate PRF based on the sub-exponential DDH assumption. The security of this PRF follows from Lemma 2.1 by an argument identical to the one in Section 3.2.

3.4 Read-once formulas

Read-once boolean formula provide a different generalization of hypercubes and they too admit an efficient aggregation algorithm for the Naor-Reingold PRF, with a similar security guarantee.

A boolean formula on ℓ variables is a circuit on $x = (x_1, \dots, x_\ell) \in \{0, 1\}^\ell$ composed of only AND, OR, and NOT gates. A *read-once boolean formula* is a boolean formula with fan-out 1, namely each input literal feeds into at most one gate, and each gate output feeds into at most one other gate.¹³ Let R_λ be the family of all read-once boolean formulas over $\ell(\lambda)$ variables. Without loss of generality, we restrict these circuits to be in a standard form: namely, composed of fan-in 2 and fan-out 1 AND and OR gates, and any NOT gates occurring at the inputs.

In this form, the circuit for any read-once boolean formula can be identified with a labelled binary tree; we identify a formula by the label of its root C_ϕ . Nodes with zero children are variables or their negation, labelled by x_i or \bar{x}_i , while all other nodes have 2 children and represent gates with fan-in 2. For such a node with label C , its children have labels C_L and C_R . Note that each child is itself a read-once boolean formula on fewer inputs, and their inputs are disjoint. Let the gate type of a node C be $\text{type}(C) \in \{\text{AND}, \text{OR}\}$.

We describe a recursive aggregation algorithm for computing products of PRF values over all accepting inputs for a given read-once boolean formula C_ϕ . Looking forward, we require the formula to be read-once in order for the recursion to be correct. The algorithm described reduces to that of Section 3.2 in the case where ϕ describes a hypercube.

3.4.1 Construction

The aggregation algorithm for read-once Boolean formulas takes as input the PRF key $\vec{K} = (K_0, \dots, K_\ell)$ and a formula $C_\phi \in R_\lambda$ where C_ϕ only reads the variables x_1, \dots, x_m for some $m \leq \ell$. We abuse notation and interpret C_ϕ to be a formula on both $\{0, 1\}^\ell$ and $\{0, 1\}^m$ in the natural way.

¹³We allow a formula to ignore some inputs variables; this enables the model to express hypercubes directly.

$$\text{AGG}_{k,\Pi}(C_\phi) = \prod_{x:C_\phi(x)=1} g^{K_0 \prod_{i \in [\ell]} K_i^{x_i}} \quad (1)$$

$$= g^{K_0 \sum_{x:C_\phi(x)=1} \prod_{i \in [\ell]} K_i^{x_i}} \quad (2)$$

$$= g^{K_0 \cdot A(C_\phi, 1) \cdot \prod_{m < j \leq \ell} (1 + K_j)} \quad (3)$$

where we define $A(C, 1) := \sum_{\{x \in \{0,1\}^m : C(x)=1\}} \prod_{i \in [m]} K_i^{x_i}$. If $A(C, 1)$ is efficiently computable, then **Aggregate** will simply compute it and return (3). To this end, we provide a recursive procedure for computing $A(C, 1)$.

Generalizing the definition for any sub-formula C with variables named x_1 to x_m , define the values $A(C, 0)$ and $A(C, 1)$:

$$A(C, b) := \sum_{\{x \in \{0,1\}^m : C(x)=b\}} \prod_{i \in [m]} K_i^{x_i}.$$

Recursively compute $A(C, b)$ as follows:

- If C is a literal for variable x_i , then by definition:

$$A(C, b) = \begin{cases} K_i & \text{if } C = x_i \\ 1 & \text{if } C = \bar{x}_i \end{cases}$$

- Else, if $\text{type}(C) = \text{AND}$: Let C_L and C_R be the children of C . By hypothesis, we can recursively compute $A(C_L, b)$ and $A(C_R, b)$ for $b \in \{0, 1\}$. Compute $A(C, b)$ as:

$$\begin{aligned} A(C, 1) &= A(C_L, 1) \cdot A(C_R, 1) \\ A(C, 0) &= A(C_L, 0) \cdot A(C_R, 0) + A(C_L, 1) \cdot A(C_R, 0) + A(C_L, 0) \cdot A(C_R, 1) \end{aligned}$$

- Else, $\text{type}(C) = \text{OR}$: Let C_L and C_R be the children of C . By hypothesis, we can recursively compute $A(C_L, b)$ and $A(C_R, b)$ for $b \in \{0, 1\}$. Compute $A(C, b)$ as:

$$\begin{aligned} A(C, 1) &= A(C_L, 1) \cdot A(C_R, 1) + A(C_L, 1) \cdot A(C_R, 0) + A(C_L, 0) \cdot A(C_R, 1) \\ A(C, 0) &= A(C_L, 0) \cdot A(C_R, 0) \end{aligned}$$

Lemma 3.6. $A(C, b)$ as computed above is equal to $\sum_{\{x \in \{0,1\}^m : C(x)=b\}} \prod_{i \in [m]} K_i^{x_i}$

Proof. For C a literal, the correctness is immediate. We must check the recursion for each $\text{type}(C) \in \{\text{AND}, \text{OR}\}$ and $b \in \{0, 1\}$. We only show the case for $b = 1$ when C is an OR gate; the other three cases can be shown similarly.

Let $S_{b_L, b_R} = \{x = (x_L, x_R) : (C_L(x_L), C_R(x_R)) = (b_L, b_R)\}$ be the set of inputs (x_L, x_R) to C such that $C_L(x_L) = b_L$ and $C_R(x_R) = b_R$. The set $\{x : C(x) = 1\}$ can be decomposed into the disjoint union $S_{0,1} \sqcup S_{1,0} \sqcup S_{1,1}$. Furthermore,

$$A(C, 1) = \sum_{x \in S_{0,1}} \prod_{i \in [m]} K_i^{x_i} + \sum_{x \in S_{1,0}} \prod_{i \in [m]} K_i^{x_i} + \sum_{x \in S_{1,1}} \prod_{i \in [m]} K_i^{x_i}$$

Because C is read-once, the sets of inputs on which C_L and C_R depend are disjoint; this implies that $A(C_L, b_L) \cdot A(C_R, b_R) = \sum_{x \in S_{b_L, b_R}} \prod_{i \in [m]} K_i^{x_i}$, yielding the desired recursion. \square

Theorem 3.7. *Let $\epsilon > 0$ be a constant, choose the security parameter $\lambda = \Omega(\ell^{1/\epsilon})$, and assume $(2^{\lambda^\epsilon}, 2^{-\lambda^\epsilon})$ -hardness of the DDH assumption. Then, the collection of functions \mathcal{F}_λ defined above is a secure aggregate PRF with respect to the subsets R_λ and the product aggregation function over the group \mathcal{G} .*

Proof. Correctness is immediate from Lemma 3.6, and Equation (3). Security follows from the decisional Diffie-Hellman assumption in much the same way it did in the case of bit-fixing functions. \square

4 Connection to Learning

4.1 Preliminaries

Notation: For a probability distribution D over a set X , we denote by $x \leftarrow D$ to mean that x is sampled according to D , and $x \leftarrow X$ to denote uniform sampling from X . For an algorithm A and a function \mathcal{O} , we denote that A has oracle access to \mathcal{O} by $A^{\mathcal{O}(\cdot)}$.

We recall the definition of a “concept class”. In this section, we will often need to explicitly reason about the representations of the concept classes discussed. Therefore we make use of the notion of a “representation class” as defined by [KV94] alongside that of concept classes. This unified formalization enables us to discuss both these traditional learning models (namely, PAC and learning with membership queries) as well as the new models we present below. Our definitions are parametrized by $\lambda \in \mathbb{N}$.¹⁴

Definition 4.1 (Representation class [KV94]). *Let $K = \{K_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of sets, where each $k \in K_\lambda$ has description in $\{0, 1\}^{s_k(\lambda)}$ for some polynomial $s_k(\cdot)$. Let $X = \{X_\lambda\}_{\lambda \in \mathbb{N}}$ be a set, where each X_λ is called a domain and each $x \in X_\lambda$ has description in $\{0, 1\}^{s_x(\lambda)}$ for some polynomial $s_x(\cdot)$. With each λ and each $k \in K_\lambda$, we associate a Boolean function $f_k : X_\lambda \rightarrow \{0, 1\}$.¹⁵ We call each such function f_k a concept, and k its index or its description. For each λ , we define the concept class $C_\lambda = \{f_k : k \in K_\lambda\}$ to be the set of all concepts with index in K_λ . We define the representation class $C = \{C_\lambda\}$ to be the union of all concept classes C_λ .*

This formalization allows us to easily associate complexity classes with concepts in learning theory. For example, to capture the set of all DNF formulas on λ inputs with size at most $p(\lambda)$ for a polynomial p , we will let $X_\lambda = \{0, 1\}^\lambda$, and $K_\lambda^{p(\lambda)}$ be the set of descriptions of all DNF formulas on λ variables with size at most $p(\lambda)$ under some reasonable representation. Then a concept $f_k(x)$ evaluates the formula k on input x . Finally, $\text{DNF}_\lambda^{p(\lambda)} = \{f_k : k \in K_\lambda^{p(\lambda)}\}$ is the concept class, and $\text{DNF}^{p(\lambda)} = \{\text{DNF}_\lambda^{p(\lambda)}\}_{\lambda \in \mathbb{N}}$. $\text{DNF}^{p(\lambda)}$ is the representation class that computes all DNF formulas on λ variables with description of size at most $p(\lambda)$ in the given representation.

As a final observation, note that a Boolean-valued PRF family $\mathcal{F} = \{\mathcal{F}_\lambda\}$ where $\mathcal{F}_\lambda = \{f_k : X_\lambda \rightarrow \{0, 1\}\}$ with keyspace $K = \{K_\lambda\}$ and domain $X = \{X_\lambda\}$ satisfies the syntax of a representation class as defined above. This formalization is useful precisely because it captures both PRF families and complexity classes, enabling lower bounds in various learning models.

In proving lower bounds for learning representation classes, it will be convenient to have a notion of containment for two representation classes.

¹⁴When clear from the context, we will omit the subscript λ .

¹⁵This association is an efficient procedure for evaluating f_k . Concretely, we might consider that there is a universal circuit F_λ such that for each λ , $f_k(\cdot) = F_\lambda(k, \cdot)$.

Definition 4.2 (\subseteq). For two representation classes $\mathcal{F} = \{\mathcal{F}_\lambda\}$ and $\mathcal{G} = \{\mathcal{G}_\lambda\}$ on the same domain $X = \{X_\lambda\}$, and with indexing sets $I = \{I_\lambda\}$ and $K = \{K_\lambda\}$ respectively, we say $\mathcal{F} \subseteq \mathcal{G}$ if for all sufficiently large λ , for all $i \in I_\lambda$, there exists $k \in K_\lambda$ such that $g_k \equiv f_i$.

Informally, if a representation class contains a PRF family, then this class is hard to MQ-learn (as in [Val84]). We apply similar reasoning to more powerful learning models. For example, if \mathcal{G} is the representation class $DNF^{p(\lambda)}$ as defined above, then $\mathcal{F} \subseteq DNF^{p(\lambda)}$ is equivalent to saying that for all sufficiently large λ , the concept class \mathcal{F}_λ can be decided by a DNF on λ inputs of $p(\lambda)$ size.

We now recall some standard definitions.

Definition 4.3 (ϵ -approximation). Let $f, h : X \rightarrow \{0, 1\}$ be arbitrary functions. We say h ϵ -approximates f if $\Pr_{x \leftarrow X}[h(x) \neq f(x)] \leq \epsilon$.

In general, ϵ -approximation is considered under a general distribution on X , but we will consider only the uniform distribution in this work.

Definition 4.4 (PAC learning). For a concept $f : X_\lambda \rightarrow \{0, 1\}$, and a probability distribution D_λ over X_λ , the example oracle $EX(f, D_\lambda)$ takes no input and returns $(x, f(x))$ for $x \leftarrow D_\lambda$. An algorithm \mathcal{A} is an (ϵ, δ) -PAC learning algorithm for representation class \mathcal{C} if for all sufficiently large λ , $\epsilon = \epsilon(\lambda) > 0$, $\delta = \delta(\lambda) > 0$ and $f \in \mathcal{C}_\lambda$,

$$\Pr[\mathcal{A}^{EX(f, D_\lambda)} = h : h \text{ is an } \epsilon\text{-approximation to } f] \geq 1 - \delta$$

Definition 4.5 (MQ learning). For a concept $f : X_\lambda \rightarrow \{0, 1\}$, the membership oracle $MEM(f)$ takes as input a point $x \in X_\lambda$ and returns $f(x)$. An algorithm \mathcal{A} is an (ϵ, δ) -MQ learning algorithm for representation class \mathcal{C} if for all sufficiently large λ , $\epsilon = \epsilon(\lambda) > 0$, $\delta = \delta(\lambda) > 0$, and $f \in \mathcal{C}_\lambda$,

$$\Pr[\mathcal{A}^{MEM(f)} = h : h \text{ is an } \epsilon\text{-approximation to } f] \geq 1 - \delta$$

We consider only PAC learning *with uniform examples*, where D_λ is the uniform distribution over X_λ . In this case, MQ is strictly stronger than PAC: everything that is PAC learnable is MQ learnable.

Observe that for any $f : X_\lambda \rightarrow \{0, 1\}$, either $h(x) = 0$ or $h(x) = 1$ will $\frac{1}{2}$ -approximate f . Furthermore, if \mathcal{A} is inefficient, f may be learned exactly. For a learning algorithm to be non-trivial, we require that it is *efficient* in λ , and that it at least *weakly* learns \mathcal{C} .

Definition 4.6 (Efficient- and weak- learning).

- \mathcal{A} is said to be *efficient* if the time complexity of \mathcal{A} and h are polynomial in $1/\epsilon, 1/\delta$, and λ .
- \mathcal{A} is said to *weakly learn* \mathcal{C} if there exist some polynomials $p_\epsilon(\lambda), p_\delta(\lambda)$ for which $\epsilon \leq \frac{1}{2} - \frac{1}{p_\epsilon(\lambda)}$ and $\delta \leq 1 - \frac{1}{p_\delta(\lambda)}$.
- We say a representation class is *learnable* if it is both *efficiently* and *weakly* learnable. Otherwise, it is *hard* to learn.

Lastly, we recall the efficiently recognizable ensembles of sets as defined in Section 2. We occasionally call such ensembles indexed, or succinct. Throughout this section, we require this property of our set ensembles \mathcal{S} . Both the MQ_{RA} and AQ learning models that we present are defined with respect to $\mathcal{S} = \{\mathcal{S}_\lambda\}$, an efficiently recognizable ensemble of subsets of the domain X_λ .

4.2 Membership queries with restriction access

In the PAC-with-Restriction Access model of learning of Dvir, et al [DRWY12], a powerful generalization of PAC learning is studied: rather than receiving random examples of the form $(x, f(x))$ for the concept f , the learning algorithm receives a random "restriction" of f - an implementation of the concept for a subset of the domain. Given this implementation of the restricted concept, the learning algorithm can both evaluate f on many related inputs, and study the properties of the restricted implementation itself. We consider an even stronger setting: instead of receiving random restrictions, the learner can adaptively request any restriction from a specified class \mathcal{S} . We call this model *membership queries with restriction access* (MQ_{RA}).

As a concrete example to help motivate and understand the definitions, we consider DNF formulas. For a DNF formula ϕ , a natural restriction might set the values of some of the variables. Consequently, some literals and clauses may have their values determined, yielding a simpler DNF formula ϕ' which agrees with ϕ on this restricted domain. This is the 'restricted concept' that the learner receives.

This model is quite powerful; indeed, decision trees and DNFs are efficiently learnable in the PAC-with-restriction-access learning model whereas neither is known to be learnable in plain PAC model [DRWY12]. Might this access model be too powerful or are there concepts that cannot be learned?

Looking forward, we will show that constrained PRFs correspond to hard-to-learn concepts in the MQ_{RA} learning model. In the remainder, we will formally define the learning model, define constrained PRFs, and prove the main lower bound of this section.

4.2.1 MQ_{RA} learning

While the original restriction access model only discusses restrictions fixing individual input bits for a circuit, we consider more general notions of restrictions.

Definition 4.7 (Restriction). *For a concept $f : X_\lambda \rightarrow \{0, 1\}$, a restriction $S \subseteq X_\lambda$ is a subset of the domain. The restricted concept $f|_S : S \rightarrow \{0, 1\}$ is equal to f on S .*

While general restrictions can be studied, we consider the setting in which all restrictions S are in a specified set of restrictions \mathcal{S} . For a DNF formula ϕ , a restriction might be $S = \{x : x_1 = 1 \wedge x_4 = 0\}$. This restriction is contained in the set of 'bit-fixing' restrictions in which individual input bits are fixed. In fact, this class of restrictions is all that is considered in [DRWY12]; we generalize their model by allowing more general classes of restrictions.

In the previous example, a restricted DNF can be naturally represented as another DNF. More generally, we allow a learning algorithm to receive representations of restricted concepts. These representations are computed according to a Simplification Rule.¹⁶

Definition 4.8 (Simplification Rule). *For each λ , let $\mathcal{C}_\lambda = \{f_k : X_\lambda \rightarrow \{0, 1\}\}_{k \in K_\lambda}$ be a concept class, \mathcal{S}_λ an efficiently recognizable ensemble of subsets of X_λ , and $S \in \mathcal{S}_\lambda$ be a restriction. A simplification of $f_k \in \mathcal{C}_\lambda$ according to S is the description $k_S \in K_\lambda$ of a concept f_{k_S} such that $f_{k_S} = f_k|_S$. A simplification rule for $\mathcal{C} = \{\mathcal{C}_\lambda\}$ and $\mathcal{S} = \{\mathcal{S}_\lambda\}$ is a mapping $\text{Simp}_\lambda : (k, S) \mapsto k_S$ for all $k \in K_\lambda, S \in \mathcal{S}_\lambda$.*

¹⁶ Whereas a DNF with some fixed input bits is naturally represented by a smaller DNF, when considering general representation classes and general restrictions, this is not always the case. Indeed, the simplification of f according to S may be in fact more complex. We use the term "Simplification Rule" for compatibility with [DRWY12].

In the PAC-learning with restriction access (PAC_{RA}) learning model considered in [DRWY12], the learner only receives random restrictions. Instead, we consider the setting where the learner can adaptively request any restriction from a specified class \mathcal{S} . This model – which we call *membership queries learning with restriction access* (MQ_{RA}) – is a strict generalization of PAC_{RA} for efficiently samplable distributions over restrictions (including all the positive results in [DRWY12]). Further observe that this strictly generalizes the membership oracle of MQ learning if \mathcal{S} is such that for each x , it is easy to find a restriction S covering x .

In traditional learning models (PAC, MQ) it is trivial to output a hypothesis that $\frac{1}{2}$ -approximates any concept f ; a successful learning algorithm is required to learn substantially more than half of the concept. With restriction queries, the learning algorithm is explicitly given the power to compute on some fraction α of the domain. Consequently, outputting an $\epsilon \geq (\frac{1-\alpha}{2})$ -approximation to f is trivial; we require a successful learning algorithm to do substantially better. This reasoning is reflected in the definition of *weak* MQ_{RA} learning below.

Definition 4.9 (Membership queries with restriction access (MQ_{RA})). *In a given execution of an oracle algorithm \mathcal{A} with access to a restriction oracle **Simp**, let $X_S \subseteq X_\lambda$ be the union of all restrictions $S \in \mathcal{S}_\lambda$ queried by \mathcal{A} . \mathcal{S} is an efficiently recognizable ensemble of subsets of the domain X_λ .*

*An algorithm \mathcal{A} is an $(\epsilon, \delta, \alpha)$ - MQ_{RA} learning algorithm for representation class \mathcal{C} with respect to a restrictions in \mathcal{S} and simplification rule **Simp** if, for all sufficiently large λ , for every $f_k \in \mathcal{C}_\lambda$, $\Pr[\mathcal{A}^{\text{Simp}(k, \cdot)} = h] \geq 1 - \delta$ where h is an ϵ -approximation to f , – and furthermore – $|X_S| \leq \alpha |X_\lambda|$.*

\mathcal{A} is said to weakly MQ_{RA} -learn if $\alpha \leq 1 - \frac{1}{p_\alpha(\lambda)}$, $\epsilon \leq (1 - \alpha)(\frac{1}{2} - \frac{1}{p_\epsilon(\lambda)})$, $\delta \leq 1 - \frac{1}{p_\delta(\lambda)}$ for some polynomials $p_\alpha, p_\epsilon, p_\delta$.

4.2.2 Constrained PRFs

We look to constrained pseudorandom functions for hard-to-learn concepts in the restriction access model. To support the extra power of the restriction access model, our PRFs will need to allow efficient evaluation on restrictions of the domain while maintaining some hardness on the remainder. Constrained PRFs [KPTZ13a, BGI14a, BW13a] provide just this power. For showing hardness of restriction access learning, the constrained keys will correspond to restricted concepts; the strong pseudorandomness property will give the hardness result.

Definition: Syntax A family of functions $\mathcal{F} = \{F_\lambda : K_\lambda \times X_\lambda \rightarrow Y_\lambda\}$ is said to be *constrained* with respect to a set system \mathcal{S} , if it supports the additional efficient algorithms:

- $\text{Constrain}_\lambda(k, S)$: A randomized algorithm, on input $(k, S) \in K_\lambda \times \mathcal{S}_\lambda$, outputs a *constrained key* k_S . We $\tilde{K}_\lambda \triangleq \text{Support}(\text{Constrain}(k, S))$ the set of all constrained keys.
- $\text{Eval}_\lambda(k_S, x)$: A deterministic algorithms taking input $(k_S, x) \in \tilde{K}_\lambda \times X_\lambda$, and satisfying the following correctness guarantee:

$$\text{Eval}(\text{Constrain}(k, S), x) = \begin{cases} F(k, x) & \text{if } x \in S \\ \perp \notin Y & \text{otherwise.} \end{cases}$$

Definition: Security Game

- \mathcal{C} picks a random key $k \in K_\lambda$ and initializes two empty subsets of the domain: $C, V = \emptyset$. C and V are subsets of X_λ which must satisfy the invariant that $C \cap V = \emptyset$. C will keep track

the inputs $x \in X_\lambda$ to the Challenge oracle, and V will be the union of all sets S queries to Constrain plus all points $x \in X_\lambda$ to the Eval oracle.

- \mathcal{C} picks $b \in \{0, 1\}$ to run $\text{EXP}(b)$, and exposes the following three oracles to \mathcal{A} :

Eval(x): On input $x \in X_\lambda$, outputs $F(k, x)$. $V \leftarrow V \cup \{x\}$.

Constrain(S): On input $S \in \mathcal{S}_\lambda$, outputs k_S . $V \leftarrow W \cup S$.

Challenge(x): On input $x \in X_\lambda$, outputs:

$$\begin{array}{ll} F(k, x) & \text{in EXP}(0) \\ y \leftarrow Y_\lambda & \text{in EXP}(1) \end{array} .$$

In $\text{EXP}(1)$, the responses to Challenge are selected uniformly at random from the range, with the requirement that the responses be consistent for identical inputs x .

- The adversary queries the oracles with the requirement that $C \cap V = \emptyset$, and outputs a bit $b' \in \{0, 1\}$.

Definition 4.10. *The advantage is defined as $\text{ADV}_\lambda^{\text{cPRF}}(\mathcal{A}) := \Pr[b' = b]$ in the above security game.*

Definition 4.11 (Constrained PRF (cPRF)). *A family of functions $\mathcal{F} = \{F_\lambda : K_\lambda \times X_\lambda \rightarrow Y_\lambda\}$ constrained with respect to \mathcal{S} is a constrained PRF if for all probabilistic polynomial-time adversaries \mathcal{A} and for all sufficiently large λ and all polynomials $p(n)$:*

$$\text{ADV}_\lambda^{\text{cPRF}}(\mathcal{A}) < \frac{1}{2} + \frac{1}{p(n)},$$

over the randomness of \mathcal{C} and \mathcal{A} .

4.2.3 Hardness of restriction access Learning

We will now prove that if a constrained PRF \mathcal{F} with respect to set system \mathcal{S} is computable in representation class \mathcal{C} , then \mathcal{C} hard to MQ_{RA} -learn with respect to \mathcal{S} and some simplification rule.

Theorem 4.1. *Let $\mathcal{F} = \{F_\lambda\}$ be a Boolean-valued constrained PRF (also interpreted as a representation class) with respect to sets \mathcal{S} and key-space K . Let $\text{EVAL} = \{\text{EVAL}_\lambda\}$ be a representation class where each EVAL_λ is defined as:*

$$\text{EVAL}_\lambda = \{g_{k_S}(\cdot) : g_{k_S}(x) = \text{PRF.Eval}(k_S, x)\}.$$

*Namely, each concept in the class EVAL_λ is indexed by $k_S \in \tilde{K}_\lambda$ and has X_λ as its domain. For any representation class $\mathcal{C} = \{\mathcal{C}_\lambda\}$ such that $\mathcal{F} \subseteq \mathcal{C}$ and $\text{EVAL} \subseteq \mathcal{C}$, there exists a simplification rule **Simp** such that \mathcal{C} is hard to MQ_{RA} -learn with respect to the set of restrictions \mathcal{S} and the simplification rule **Simp**.*

Existing constructions of constrained PRFs [BW13a] yield the following corollaries:

Corollary 4.2. *Let $n = n(\lambda)$ be a polynomial, and assume that for the $n + 1$ -MDDH problem, every adversary time $\text{poly}(\lambda)$ the advantage is at most $\epsilon(\lambda)/2^n$. Then there is a simplification rule such that NC^1 is hard to MQ_{RA} -learn with respect to restrictions in \mathcal{HC} ¹⁷.*

Corollary 4.3. *Assuming the existence of one-way functions, there is a simplification rule such that P/poly is hard to MQ_{RA} -learn with respect to restrictions in $\mathcal{S}_{[a,b]}$ ¹⁸.*

Remarks: The Simplification Rule here is really the crux of the issue. In our theorem, *there exists* a simplification rule under which we get a hardness result. This may seem somewhat artificial. On the other hand, this implies that the restriction-access learnability (whether PAC- or MQ-RA) of a concept class crucially depends on the simplification rule, as the trivial simplification rule of $\text{Simp}(k, S) = k$ admits a trivial learning-algorithm in either setting. This work reinforces that the choice simplification rule can affect the learnability of a given representation class. Positive results for restriction access learning that were independent of the representation would be interesting.

Proof of Theorem 4.1. We interpret $\mathcal{F} = \{\mathcal{F}_\lambda\}$ as a representation class. For each λ , the concepts $f_k \in \mathcal{F}_\lambda$ are indexed by K_λ and have domain X_λ . Let $EVAL = \{EVAL_\lambda\}$ be a representation class defined as in the theorem statement. The indexing set for $EVAL_\lambda$ is \tilde{K}_λ , the set of constrained keys k_S for $k \in K_\lambda, S \in \mathcal{S}_\lambda$.

Let $\mathcal{C} = \{\mathcal{C}_\lambda\}$ be a representation class, with domain X_λ and indexing set I_λ . For $i \in I_\lambda$, c_i is a concept in \mathcal{C}_λ .

By hypothesis, $\mathcal{F} \subseteq \mathcal{C}$: for sufficiently large λ , for all $k \in K_\lambda$ there exists $i \in I_\lambda$ such that $c_i \equiv f_k$. Similarly, for all $k_S \in \tilde{K}_\lambda$ there exists $i \in I_\lambda$ such that $c_i \equiv \text{Eval}_\lambda(k_S, \cdot)$. For concreteness, let M_λ be this map from $K_\lambda \cup \tilde{K}_\lambda$ to I_λ .¹⁹

We can now specify the simplification rule $\text{Simp}_\lambda : I_\lambda \times \mathcal{S}_\lambda \rightarrow I_\lambda$. Letting $M_\lambda(K_\lambda) \subseteq I_\lambda$ be the image of K_λ under M_λ :

$$\text{Simp}_\lambda(i, S) = \begin{cases} M_\lambda(\text{Constrain}_\lambda(M_\lambda^{-1}(i), S)) & \text{if } i \in M_\lambda(K_\lambda) \\ i & \text{otherwise.} \end{cases}$$

For example, i may be a circuit computing the PRF f_k for some $k = M_\lambda^{-1}(i)$. The simplification computes the circuit corresponding to a constrained PRF key, if the starting circuit already computes a member of the PRF family \mathcal{F}_λ .²⁰

Reduction: Suppose, for contradiction, that there exists an such an efficient learning algorithm \mathcal{A} for \mathcal{C} as in the statement of the theorem. We construct algorithm \mathcal{B} breaking the constrained PRF security. In the PRF security game, \mathcal{B} is presented with the oracles $f_k(\cdot)$, $\text{Constrain}_\lambda(k, \cdot)$, and $\text{Challenge}_\lambda(\cdot)$, for some $k \leftarrow K_\lambda$. Run \mathcal{A} , and answer queries $S \in \mathcal{S}_\lambda$ to the restriction oracle by querying $\text{Constrain}_\lambda(k, S)$, receiving k_S , and returning $M_\lambda(k_S)$. Once \mathcal{A} terminates, it outputs hypothesis h . By assumption on \mathcal{A} , with probability at least $1 - \delta > \frac{1}{p_\delta(\lambda)}$, the hypothesis h is an ϵ -approximation of $c_{M(k)} \equiv f_k$ with $\epsilon \leq \frac{1-\alpha}{2}$ and $\alpha < 1 - \frac{1}{p_\alpha(\lambda)}$.

After receiving hypothesis h , \mathcal{B} estimates the probability $\Pr_{x \leftarrow X \setminus X_S}[h(x) = \text{Challenge}_\lambda(x)]$. In $\text{EXP}(0)$, this probability is at least $1 - \epsilon$ with probability at least $1 - \delta$; in $\text{EXP}(1)$, it is exactly $1/2$.

¹⁷as defined in Section 3.

¹⁸as defined in Section 3.

¹⁹ This is a non-uniform reduction.

²⁰Note that while the inverse map M_λ^{-1} may be inefficient, in our reduction, the concept in question is represented by a PRF key k . Thus \mathcal{B} must only compute the forward map M_λ .

To sample uniform $x \in X \setminus X_S$, we simply take a uniform $x \in X$: with probability $1 - \alpha \geq 1/p_\alpha(n)$, $x \in X \setminus X_S$. Thus, \mathcal{B} runs in expected polynomial time. If the estimate is close to ϵ , guess $\text{EXP}(0)$; otherwise, flip a fair coin $b' \in \{0, 1\}$ and guess $\text{EXP}(b')$. The advantage $\text{ADV}_\lambda^{\text{cPRF}}$ of \mathcal{B} in the PRF security game is at least $\frac{1}{3p_\delta(\lambda)}$ for all sufficiently large λ (see Analysis for details), directly violating the security of \mathcal{F} .

Analysis: Let $p_b \triangleq \Pr_{x \in X \setminus X_S}[h(x) \neq \text{Challenge}_\lambda(x) | \text{EXP}(b)]$ be the probability taken with respect to experiment $\text{EXP}(b)$. In $\text{EXP}(1)$, Challenge_λ is a uniformly random function. Thus, $p_1 = \frac{1}{2}$. With high probability, \mathcal{B} will output a random bit $b' \in \{0, 1\}$, guessing correctly with probability $1/2$.

In $\text{EXP}(0)$, h is an ϵ -approximation to f_k , and thus to Challenge_λ , with probability at least $1 - \delta$. In this case, $p_0 \geq 1 - \epsilon \geq \frac{1}{2} + \frac{1}{p_\epsilon(\lambda)}$. By a Hoeffding bound, \mathcal{B} will guess $b' = 0$ with high probability by estimating p using only polynomial in $\lambda, p_\epsilon(\lambda)$ samples. On the other hand, if h is not an ϵ -approximation, \mathcal{B} will $b' = 0$ with probability at least $1/2$.

Let $\text{negl}(\lambda)$ be the error probability from the Hoeffding bound, which can be made exponentially small in λ . The success probability is: $\Pr[b = b' | b = 0] \geq (1 - \delta)(1 - \text{negl}(\lambda)) + \frac{\delta}{2}$ which, for $1 - \delta \geq \frac{1}{p_\delta(\lambda)}$ is at least $\frac{1}{3p_\delta(\lambda)} + \frac{1}{2}$ for sufficiently large λ . Thus \mathcal{B} a non-negligible advantage of $1/3p_\delta(\lambda)$ in the constrained PRF security game. \square

4.3 Learning with related concepts

The idea that some functions or concepts are related to one another is very natural. For a DNF formula, for instance, related concepts may include formulas where a clause has been added or formulas where the roles of two variables are swapped. For a decision tree, we could consider removing some accepting leaves and examining the resulting behavior. We might consider a circuit; related circuits might alter internal gates or fix the values of specific input or internal wires.

Formally, we consider indexed representation classes. As discussed in the preliminaries, general classes of functions are easily represented as an indexed family. For example, we may consider the bit representation of a function (say, a log-depth circuit) as an index into a whole class (NC^1). This formalism enables the study of related concepts by instead considering concepts whose keys are related in some way. The related concept setting shares an important property with the restriction access setting: different representations of the same functions might have very different properties. Exploring the properties of different representations – and perhaps their RC learnability as defined below – is a direction for future work.

In our model of learning with related concepts, we allow the learner to query a membership oracle for the concept $f_k \in \mathcal{C}_\lambda$ and also for some ‘related’ concepts $f_{\phi(k)} \in \mathcal{C}_\lambda$ for some functions ϕ . The *related-concept deriving (RCD)* function ϕ is restricted to be from a specified class, Φ_λ . For each $\phi \in \Phi_\lambda$, a learner can access the membership oracle for $f_{\phi(k)}$. For example: let $K_\lambda = \{0, 1\}^\lambda$ and let

$$\Phi_\lambda^\oplus = \{\phi_\Delta : k \mapsto k \oplus \Delta\}_{\Delta \in \{0,1\}^\lambda} \quad (4)$$

Definition 4.12 (Φ -Related-Concept Learning Model). *For \mathcal{C} a representation class indexed by $\{K_\lambda\}$, let $\Phi = \{\Phi_\lambda\}$, with each $\Phi_\lambda = \{\phi : K_\lambda \rightarrow K_\lambda\}$ a set of functions on K_λ containing the identity function id_λ . The related-concept oracle RC_k , on query (ϕ, x) , responds with $f_{\phi(k)}(x)$, for all $\phi \in \Phi_\lambda$ and $x \in X_\lambda$.*

An algorithm A is an (ϵ, δ) - Φ -RC learning algorithm for a \mathcal{C} if, for all sufficiently large λ , for every $k \in K_\lambda$, $\Pr[A^{\text{RC}_k(\cdot, \cdot)} = h] \geq 1 - \delta$ where h is an ϵ -approximation f_k .

Studying the related-concept learnability of standard representation classes (ex: DNFs and decision trees) under different RCD classes Φ is an interesting direction for future study.

4.3.1 RKA PRFs

Again we look to pseudorandom functions for hard-to-learn concepts. To support the extra power of the related concept model, our PRFs will need to maintain their pseudorandomness even when the PRF adversary has access to the function computed with related keys. Related-key secure PRFs [BC10, ABPP14] provide just this guarantee. As in the definition of RC learning, the security of related-key PRFs is given with respect to a class Φ of related-key deriving functions. As we describe in the remainder of the section, related-key secure PRFs prove hard to weakly Φ -RC learn.

Definition: Security Game

Let $\Phi_\lambda \subseteq \text{Fun}(K_\lambda, K_\lambda)$ be a subset of functions on K_λ . The set $\Phi = \{\Phi_\lambda\}$ is called the *Related-Key Deriving (RKD)* class and each function $\phi \in \Phi_\lambda$ is an RKD function.

- \mathcal{C} picks a random key $k \in K_\lambda$, a bit $b \in \{0, 1\}$, and exposes the oracle according to $\text{EXP}(b)$:
 $\text{RKFn}_\lambda(\phi, x)$: On input $(\phi, x) \in \Phi_\lambda \times X_\lambda$, outputs:

$$\begin{array}{ll} F(\phi(k), x) & \text{in EXP}(0) \\ y \leftarrow Y_\lambda & \text{in EXP}(1) \end{array} .$$

In $\text{EXP}(1)$, the responses to RKFn_λ are selected uniformly at random from the range, with the requirement that the responses be consistent for identical inputs (ϕ, x) .

- The adversary interacts with the oracle, and outputs a bit $b' \in \{0, 1\}$.

Definition 4.13. *The advantage is defined as $\text{ADV}_\lambda^{\Phi\text{-RKA}}(\mathbf{A}) := \Pr[b' = b]$ in the above security game.*

Definition 4.14 (Φ Related-key attack PRF (Φ -RKA-PRF)). *Let $\mathcal{F} = \{F_\lambda : K_\lambda \times X_\lambda \rightarrow Y_\lambda\}$ be family of functions and let $\Phi = \{\Phi_\lambda\}$ with each $\Phi_\lambda \subseteq \text{Fun}(K_\lambda, K_\lambda)$ be a set of functions on K_λ . \mathcal{F} is a Φ related-key attack PRF family if for all probabilistic polynomial-time adversaries \mathbf{A} and for all sufficiently large λ and all polynomials $p(n)$:*

$$\text{ADV}_\lambda^{\Phi\text{-RKA}}(\mathbf{A}) < \frac{1}{2} + \frac{1}{p(n)},$$

over the randomness of \mathcal{C} and \mathbf{A} .

4.3.2 Hardness of related concept learning

In the Appendix C, we present a concept that can be RC-learned under Φ^\oplus (Equation 4), but is hard to weakly learn with access to membership queries. We construct the concept \mathcal{F} from a PRF \mathcal{G} and a PRP P . Informally, the construction works by hardcoding the the PRF key in the function values on a related PRF. With the appropriate related-concept access, a learner can learn the PRF key.

We now present a general theorem relating RKA-PRFs to hardness of RC learning. This connection yields hardness for a class \mathcal{C} with respect to restricted classes of relation functions Φ . More general hardness results will require new techniques.

Theorem 4.4. *Let \mathcal{F} be a boolean-valued Φ -RKA-PRF with respect to related-key deriving class Φ and keyspace K . For a representation class \mathcal{C} , if $\mathcal{F} \subseteq \mathcal{C}$, then there exists an related-concept deriving class Ψ such that \mathcal{C} is hard to Ψ -RC.*

As a corollary, we get a lower bound coming from the RKA-PRF literature. For a group $(G, +)$, and $K = G^m$, define the the element-wise addition RKD functions as

$$\Phi_+^m = \{\phi_\Delta : k[1], \dots, k[m] \mapsto k[1] + \Delta[1], \dots, k[m] + \Delta[m]\}_{\Delta \in G^m} \quad (5)$$

Notice that Φ_+^m directly generalizes Φ^\oplus with $G = \mathbb{Z}_2$. For this natural RKD function family, we are able to provide a strong lower bound based on the hardness of DDH and the existence of collision-resistant hash functions using the RKA-PRF constructions from [ABPP14].

Corollary 4.5 (Negative Result from RKA-PRF). *If the DDH assumption holds and collision-resistant hash functions exist NC^1 is hard to Φ_+^m -RKA-learn.*

Proof of Theorem 4.4. We interpret $\mathcal{F} = \{\mathcal{F}_\lambda\}$ as a representation class. For each λ , the concepts $f_k \in \mathcal{F}_\lambda$ are indexed by K_λ and have domain X_λ . Let $\mathcal{C} = \{\mathcal{C}_\lambda\}$ be a representation class, with domain X_λ and indexing set I_λ . For $i \in I_\lambda$, c_i is a concept in \mathcal{C}_λ .

By hypothesis, $\mathcal{F} \subseteq \mathcal{C}$: for sufficiently large λ , for all $k \in K_\lambda$ there exists $i \in I_\lambda$ such that $c_i \equiv f_k$. For concreteness, let M_λ be this map from K_λ to I_λ .²¹

We can now specify the RCD class $\Psi_\lambda : I_\lambda \rightarrow I_\lambda$. Let $M_\lambda(K_\lambda) \subseteq I_\lambda$ be the image of K_λ under M_λ . We define $\Psi_\lambda = \{\psi_\phi : \phi \in \Phi_\lambda\}$:

$$\psi_\phi(i) = \begin{cases} M_\lambda \circ \phi \circ M_\lambda^{-1}(i) & \text{if } i \in M_\lambda(K_\lambda) \\ i & \text{otherwise.} \end{cases}$$

Reduction: Suppose, for contradiction, that there exists an efficient Ψ -RC learning algorithm \mathcal{A} for \mathcal{C} as in the statement of the theorem. We construct algorithm \mathcal{B} breaking the Φ -RKA-PRF security of \mathcal{F} . In the PRF security game, \mathcal{B} is presented with the oracle $RKFn(\cdot, \cdot)$; \mathcal{A} is presented with the oracle $RC(\cdot, \cdot)$. Run \mathcal{A} , and answer queries $(\psi_\phi, x) \in \Psi_\lambda \times X_\lambda$ to RC by querying $RKFn$ on (ϕ, x) and passing the response along to \mathcal{A} . Let $X_\mathcal{A} = \{x \in X_\lambda : \mathcal{A} \text{ queried } (\psi, x) \text{ for some } \psi\}$. Once \mathcal{A} terminates, it outputs hypothesis h . In $\text{EXP}(0)$, $RKFn()$ responds according to f_k for some $k \in K_\lambda$; in this case, \mathcal{B} simulates the RC oracle for the concept $c_{M(k)}$.

After receiving hypothesis h , \mathcal{B} estimates the probability $\Pr_{x \leftarrow X \setminus X_\mathcal{A}}[h(x) = RKFn_\lambda(x)]$. In $\text{EXP}(0)$, this probability is at least $1 - \epsilon$ with probability at least $1 - \delta$; in $\text{EXP}(1)$, it is exactly $1/2$. To sample uniform $x \in X \setminus X_\mathcal{A}$, we simply take a uniform $x \in X$: with high probability $x \in X \setminus X_\mathcal{A}$. If the estimate is close to ϵ , guess $\text{EXP}(0)$; otherwise, flip an fair coin $b' \in \{0, 1\}$ and guess $\text{EXP}(b')$. The advantage $ADV_\lambda^{\Phi\text{-RKA}}$ of \mathcal{B} in the PRF security game is at least $\frac{1}{3p_\delta(n)}$ (see Analysis for details) for all sufficiently large λ , directly violating the security of \mathcal{F} .

Analysis: Let $p_b \triangleq \Pr_{x \in X \setminus X_\mathcal{A}}[h(x) \neq RKFn(\text{id}_\lambda, x) | \text{EXP}(b)]$ be the probability taken with respect to experiment $\text{EXP}(b)$. In $\text{EXP}(1)$, $RKFn$ is a uniformly random function. Thus, $p_1 = \frac{1}{2}$. With high probability, \mathcal{B} will output a random bit $b' \in \{0, 1\}$, guessing correctly with probability $1/2$.

In $\text{EXP}(0)$, h is an ϵ -approximation to $RKFn(\text{id}, \cdot)$ with probability at least $1 - \delta$. In this case, $p_0 \geq 1 - \epsilon \geq \frac{1}{2} + \frac{1}{p_\epsilon(\lambda)}$. By a Hoeffding bound, \mathcal{B} will guess $b' = 0$ with high probability by estimating

²¹ This is a non-uniform reduction in general, but in most cases, the map M is known. That is, M_λ is the map that takes a key and outputs a circuit computing the function.

p using only polynomial in $\lambda, p_\epsilon(\lambda)$ samples. On the other hand, if h is not an ϵ -approximation, \mathcal{B} will $b' = 0$ with probability at least $1/2$.

Let $\text{negl}(\lambda)$ be the error probability from the Hoeffding bound, which can be made exponentially small in λ . The success probability is: $\Pr[b = b' | b = 0] \geq (1 - \delta)(1 - \text{negl}(\lambda)) + \frac{\delta}{2}$ which, for $1 - \delta \geq \frac{1}{p_\delta(\lambda)}$ is at least $\frac{1}{3p_\delta(\lambda)} + \frac{1}{2}$ for sufficiently large λ . Thus \mathcal{B} a non-negligible advantage of $1/3p_\delta(\lambda)$ in the Φ -RKA-PRF security game. \square

Proof. For $n \in \mathbb{N}$ let $\mathbb{G} = \langle g \rangle$ be a group of prime order $p = p(n)$, $X_n = \{0, 1\}^{m(n)} \setminus \{0^n\}$, $K_n = \mathbb{Z}_p^m(n)$, and define $F_k(x)$ as in Theorem 4.5 of [Abdalla] (). Let ϕ_+^m be as above over \mathbb{K} . \square

4.4 Learning with Aggregate Queries

This computational learning model is inspired by our aggregate PRFs. Rather than being a natural model in its own right, this model further illustrates how cryptography and learning are in some senses duals. Here, we consider a new extension to the power of the learning algorithm. Whereas membership queries are of the form “What is the label of an example x ?”, we grant the learner the power to request the evaluation of simple functions on tuples of examples (x_1, \dots, x_k) such as “How many of $(x_1 \dots x_k)$ are in C ?” or “Compute the product of the labels of (x_1, \dots, x_k) ?”. Clearly, if k is polynomial then this will result only a polynomial gain in the query complexity of a learning algorithm in the best case. Instead, we propose to study cases when k may be super polynomial, but the description of the tuples is succinct. For example, the learning algorithm might query the number of x ’s in a large interval that are positive examples in the concept.

As with the restriction access and related concept models – and the aggregate PRFs we define in this work – the Aggregate Queries (AQ) learning model will be considered with restrictions to both the types of aggregate functions Γ the learner can query, and the sets \mathcal{S} over which the learner may request these functions to be evaluated on. We now present the AQ learning model informally:

Definition 4.15 ((Γ, \mathcal{S})-Aggregate Queries (AQ) Learning). *Let \mathcal{C} be a representation class with domains $X = \{X_\lambda\}$, and $\mathcal{S} = \{\mathcal{S}_\lambda\}$ where each \mathcal{S}_λ is a collection of efficiently recognizable subsets of the X_λ . $\Gamma : \{0, 1\}^* \rightarrow V_\lambda$ be an aggregation function [as in def.]. Let $\text{AGG}_k^\lambda \triangleq \text{AGG}_{f_k, \mathcal{S}_\lambda, \Gamma_\lambda}^\lambda$ be the aggregation oracle for $f_k \in \mathcal{C}_\lambda$, for $S \in \mathcal{S}_\lambda$ and Γ_λ .*

An algorithm \mathcal{A} is an (ϵ, δ) -(Γ, \mathcal{S})-AQ learning algorithm for \mathcal{C} if, for all sufficiently large λ , for every $f_k \in \mathcal{C}_\lambda$, $\Pr[\mathcal{A}^{\text{MEM}_{f_k}(\cdot), \text{AGG}_{f_k}^\lambda(\cdot)} = h] \geq 1 - \delta$ where h is an ϵ -approximation to f_k .

4.4.1 Hardness of aggregate query learning

Theorem 4.6. *Let \mathcal{F} be a boolean-valued aggregate PRF with respect to set system $\mathcal{S} = \{\mathcal{S}_\lambda\}$ and accumulation function $\Gamma = \{\Gamma_\lambda\}$. For a representation class \mathcal{C} , if $\mathcal{F} \subseteq \mathcal{C}$, then \mathcal{C} is hard to (Γ, \mathcal{S}) -AQ learn.*

Looking back to our constructions of aggregate pseudorandom function families from the prequel, we have the following corollaries.

Corollary 4.7. *The existence of one-way functions implies that P/poly is hard to $(\sum, \mathcal{S}_{[a,b]})$ -AQ learn, with $\mathcal{S}_{[a,b]}$ the set of sub-intervals of the domain as defined in Section 3.*

Corollary 4.8. *The DDH Assumption implies that NC^1 is hard to $(\sum, \mathcal{S}_{[a,b]})$ -AQ learn, with $\mathcal{S}_{[a,b]}$ the set of sub-intervals of the domain as defined in Section 3.*

Corollary 4.9. *The subexponential DDH Assumption implies that NC^1 is hard to (\prod, \mathcal{R}) -AQ learn, with \mathcal{R} the set of read-once boolean formulas defined in Section 3.*

Proof of Theorem 4.6. Interpreting \mathcal{F} itself as a concept class, we will show an efficient reduction from violating the pseudorandomness property of \mathcal{F} to weakly (Γ, \mathcal{S}) -AQ learning \mathcal{F} . By assumption, $\mathcal{F} \subseteq \mathcal{C}$, implying that \mathcal{C} is hard to learn as well.

Reduction: Suppose for contradiction that there exists an efficient weak learning algorithm \mathcal{A} for \mathcal{F} . We define algorithm \mathcal{B} violating the aggregate PRF security of \mathcal{F} . In the PRF security game, \mathcal{B} is presented with two oracles: $F(\cdot)$ and AGG_F^λ for a function F chosen according to the secret bit $b \in \{0, 1\}$. In EXP(0), $F = f_k$ for random $k \in K_\lambda$; by assumption $f_k \in \mathcal{C}_\lambda$. In EXP(1), F is a uniformly random function from X to $\{0, 1\}$. The learning algorithm \mathcal{A} is presented with precisely the same oracles. \mathcal{B} runs \mathcal{A} , simulating its oracles by passing queries and responses to its own oracles. $X_{\mathcal{A}} = \{x \in X_\lambda : \mathcal{A} \text{ queried } (\psi, x) \text{ for some } \psi\}$. Once \mathcal{A} terminates, it outputs hypothesis h .

After receiving hypothesis h , \mathcal{B} estimates the probability

$$p = \Pr_{x \leftarrow X \setminus X_{\mathcal{A}}} [h(x) = F(x)]$$

(using polynomial in $\lambda, p_\epsilon(\lambda)$ samples). In EXP(0), this probability is at least $1 - \epsilon$ with probability at least $1 - \delta$; in EXP(1), it is exactly $1/2$. To sample uniform $x \in X \setminus X_{\mathcal{A}}$, we simply take a uniform $x \in X$: with high probability $x \in X \setminus X_{\mathcal{A}}$. If the estimate is close to ϵ , guess EXP(0); otherwise, flip a fair coin $b' \in \{0, 1\}$ and guess EXP(b'). The advantage ADV_λ^{APRF} of \mathcal{B} in the PRF security game is at least $\frac{1}{3p_\delta(n)}$ for all sufficiently large λ (as shown below), directly violating the security of \mathcal{F} .

Let

$$p_b \triangleq \Pr_{x \in X \setminus X_{\mathcal{A}}} [h(x) \neq F(x) | EXP(b)]$$

be the probability taken with respect to experiment EXP(b). In EXP(1), F is a uniformly random function. Thus, $p_1 = \frac{1}{2}$. With high probability, \mathcal{B} will output a random bit $b' \in \{0, 1\}$, guessing correctly with probability $1/2$.

In EXP(0), h is an ϵ -approximation to F with probability at least $1 - \delta$. In this case, $p_0 \geq 1 - \epsilon \geq \frac{1}{2} + \frac{1}{p_\epsilon(\lambda)}$. By a Hoeffding bound, \mathcal{B} will guess $b' = 0$ with high probability by estimating p using only polynomial in $\lambda, p_\epsilon(\lambda)$ samples. On the other hand, if h is not an ϵ -approximation, \mathcal{B} will $b' = 0$ with probability at least $1/2$.

Let $\text{negl}(\lambda)$ be the error probability from the Hoeffding bound, which can be made exponentially small in λ . The success probability is:

$$\Pr[b = b' | b = 0] \geq (1 - \delta)(1 - \text{negl}(\lambda)) + \frac{\delta}{2}$$

which, for $1 - \delta \geq \frac{1}{p_\delta(\lambda)}$ is at least $\frac{1}{3p_\delta(\lambda)} + \frac{1}{2}$ for sufficiently large λ . Thus \mathcal{B} a non-negligible advantage of $1/3p_\delta(\lambda)$ in the (Γ, \mathcal{S}) -aggregate-PRF security game. \square

4.4.2 Acknowledgements

Aloni Cohen's research was supported in part by NSF Graduate Student Fellowship and NSF grants CNS1347364, CNS1413920 and FA875011-20225.

Shafi Goldwasser’s research was supported in part by NSF Grants CNS1347364, CNS1413920 and FA875011-20225.

Vinod Vaikuntanathan’s research was supported by DARPA Grant number FA8750-11-2-0225, an Alfred P. Sloan Research Fellowship, an NSF CAREER Award CNS-1350619, NSF Frontier Grant CNS-1414119, a Microsoft Faculty Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.

References

- [ABPP14] Michel Abdalla, Fabrice Benhamouda, Alain Passelègue, and Kenneth G. Paterson. Related-key security for pseudorandom functions beyond the linear barrier. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 77–94. Springer, 2014.
- [BC10] Mihir Bellare and David Cash. Pseudorandom functions and permutations provably secure against related-key attacks. In Tal Rabin, editor, *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*, pages 666–684. Springer, 2010.
- [BF02] Nader H Bshouty and Vitaly Feldman. On using extended statistical queries to avoid membership queries. *The Journal of Machine Learning Research*, 2:359–395, 2002.
- [BGI14a] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Krawczyk [Kra14], pages 501–519.
- [BGI14b] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Krawczyk [Kra14], pages 501–519.
- [BGV11] Siavosh Benabbas, Rosario Gennaro, and Yevgeniy Vahlis. Verifiable delegation of computation over large datasets. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 111–131. Springer, 2011.
- [BK03] Mihir Bellare and Tadayoshi Kohno. A theoretical treatment of related-key attacks: Rka-prps, rka-prfs, and applications. In *Advances in Cryptology EUROCRYPT 2003*, pages 491–506. Springer, 2003.
- [BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 410–428. Springer, 2013.

- [BW04] Andrej Bogdanov and Hoeteck Wee. A stateful implementation of a random function supporting parity queries over hypercubes. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2004, and 8th International Workshop on Randomization and Computation, RANDOM 2004, Cambridge, MA, USA, August 22-24, 2004, Proceedings*, pages 298–309, 2004.
- [BW13a] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Sako and Sarkar [SS13], pages 280–300.
- [BW13b] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Sako and Sarkar [SS13], pages 280–300.
- [DRWY12] Zeev Dvir, Anup Rao, Avi Wigderson, and Amir Yehudayoff. Restriction access. In Shafi Goldwasser, editor, *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 19–33. ACM, 2012.
- [GGI⁺02] Anna C. Gilbert, Sudipto Guha, Piotr Indyk, Yannis Kotidis, S. Muthukrishnan, and Martin Strauss. Fast, small-space algorithms for approximate histogram maintenance. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 389–398, 2002.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. Extended abstract in FOCS 84.
- [GGN10] Oded Goldreich, Shafi Goldwasser, and Asaf Nussboim. On the implementation of huge random objects. *SIAM J. Comput.*, 39(7):2761–2822, 2010.
- [HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.
- [KPTZ13a] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Sadeghi et al. [SGY13], pages 669–684.
- [KPTZ13b] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Sadeghi et al. [SGY13], pages 669–684.
- [Kra14] Hugo Krawczyk, editor. *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, volume 8383 of *Lecture Notes in Computer Science*. Springer, 2014.
- [KV94] Michael J. Kearns and Leslie G. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.
- [NR04] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudorandom functions. *J. ACM*, 51(2):231–262, 2004.

- [RR97] Alexander A. Razborov and Steven Rudich. Natural proofs. *J. Comput. Syst. Sci.*, 55(1):24–35, 1997.
- [SGY13] Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors. *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*. ACM, 2013.
- [SS13] Kazue Sako and Palash Sarkar, editors. *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, volume 8270 of *Lecture Notes in Computer Science*. Springer, 2013.
- [SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.
- [Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
- [VV86] Leslie G. Valiant and Vijay V. Vazirani. NP is as easy as detecting unique solutions. *Theor. Comput. Sci.*, 47(3):85–93, 1986.

A Simple Positive Results

In the following, we present examples of concept classes separating the Related Concept and Aggregate Query learning models from learning with Membership Queries. We emphasize that the learnability of many traditional concept classes in these models has not been studied, and more general positive results may exist. In order to exhibit separations, we present generic, contrived constructions from simple cryptographic primitives to exhibit our separations. In each case, a MQ learner cannot succeed better than a trivial algorithm, while the stronger model manages to exactly, and properly learn the function.

A.1 Related-concept

While some existing pseudorandom functions are known to suffer from related-key attacks [BK03], these vulnerabilities do not seem directly useful for a proper learning algorithm. Instead we construct a family of PRFs for which the secret key can be recovered under related-key attacks.

We demonstrate a concept that can be RC-learned under additive Φ (defined below), but is hard to weakly learn with access to membership queries. We construct the concept \mathcal{F} from a PRF \mathcal{G} and a PRP P . Informally, the construction works by hardcoding the PRF key in the function values *under a related PRF key*. With the appropriate related-key access, a learner can learn the PRF key.

Let $\mathcal{G} = \{G_k : \mathbb{Z}_{2^\lambda} \rightarrow \{0, 1\}\}_{k \in K}$ be a PRF with keyspace $K = \{0, 1\}^\lambda$ and let $P = \{\pi : K \rightarrow K\}$ be a pseudorandom permutation family on K . For each $g_k \in \mathcal{G}_K$ and $\pi \in P$, we define the following

function:

$$F_{k,\pi} = \begin{cases} x\text{th bit of } (\pi(k) \oplus k) & \text{if } x \in [0, \lambda - 1] \\ (x - \lambda)\text{th bit of } \pi^{-1}(k) & \text{if } x \in [\lambda, 2\lambda - 1] \\ g_k(x) & \text{otherwise} \end{cases}$$

Let $\mathcal{F} = \{F_{k,\pi} : k \in K, \pi \in P\}$. We interpret \mathcal{F} as a keyed concept with elements indexed by a pairs (k, π) .

We need to choose a RKD class Φ that will enable recovery of the PRF key k by accessing the PRF for key $\pi(k) \oplus k$. We choose $\Phi = \Phi^\oplus$ from Section 4.3.1:

$$\Phi^\oplus = \{\phi_\Delta : k \mapsto k \oplus \Delta\}_{\Delta \in K}$$

Note that in that section, we prove a negative result for a strictly stronger RC adversary, but with a different concept class.

Theorem A.1 (Separating RC and MQ). *The keyed concept \mathcal{F} defined above can be (efficiently) exactly Φ^\oplus -RC-learned, but is hard to even weakly MQ learn efficiently.*

Proof. Let $F_{k,\pi} \in \mathcal{F}_n$.

Φ^\oplus -RC Learning: Let $RC_{k,\pi}$ be the related-concept oracle, taking queries $(\phi, x) \in \Phi^\oplus \times \mathbb{Z}_{2\lambda}$ and returning $F_{\phi(k),\pi}(x)$. Define $\Delta \in K$ such that $\Delta[i] = F_{k,\pi}(i)$ for all $i \in [\lambda - 1]$; compute the i th bit by querying the oracle at (id, i) , where $\text{id} = 0^\lambda$ is the identity function. By construction, $k \oplus \Delta = \pi(k)$. Let $k' \in K$ such that $k'[i] = F_{\pi(k),i+\lambda}$ for all $i \in [\lambda - 1]$; we find bit $k'[i]$ by querying $(\phi_\Delta, i + \lambda)$. By construction, $k' = \pi^{-1}(\pi(k)) = k$. Given the PRF key k , we may compute $F_{k,\pi}$ on all inputs in $X \setminus [2\lambda - 1]$; simply querying those remaining points yields an exact characterization of $F_{k,\pi}$.

MQ Learning: (Informally) Given a weakly-MQ learning algorithm \mathcal{A} for \mathcal{F} , an algorithm \mathcal{B} violating the security of the pseudorandom function can be constructed. By assumption, \mathcal{A} is an (ϵ, δ) -MQ learning algorithm with ϵ and $1 - \delta$ both non-negligible in n . First, observe that \mathcal{A} is an (ϵ', δ') -MQ-learning algorithm for the following concept class, indexed by $k \in K$ and uniformly random $r_1 \in \{0, 1\}^\lambda$, with $\epsilon' \geq \epsilon - \text{negl}(\lambda)$ and $\delta' \geq \delta - \text{negl}(\lambda)$:

$$F_{k,r_1}^1 = \begin{cases} x\text{th bit of } r_1 & \text{if } x \in [0, \lambda - 1] \\ (x - \lambda)\text{th bit of } \pi^{-1}(k) & \text{if } x \in [\lambda, 2\lambda - 1] \\ g_k(x) & \text{otherwise} \end{cases}$$

Otherwise, the quality of the hypothesis output by \mathcal{A} would be noticeably different for random functions $F_{k,\pi}$ and F_{k,r_1} . By the security of the pseudorandom permutation, $\pi(k) \oplus k$ should be indistinguishable from uniformly random r_1 ; this difference could be used to violate the security of the pseudorandom permutation π .

A similar argument will show that \mathcal{A} is an (ϵ'', δ'') -MQ-learning algorithm for the following concept class, indexed by $k \in K$ and $r_1, r_2 \in \{0, 1\}^\lambda$, with $\epsilon'' \geq \epsilon' - \text{negl}(\lambda)$ and $\delta'' \geq \delta' - \text{negl}(\lambda)$:

$$F_{k,r_1}^2 = \begin{cases} x\text{th bit of } r_1 & \text{if } x \in [0, \lambda - 1] \\ (x - \lambda)\text{th bit of } r_2 & \text{if } x \in [\lambda, 2\lambda - 1] \\ g_k(x) & \text{otherwise} \end{cases}$$

Furthermore, weak learning of this concept requires weak learning of this concept even when restricting the domain to require $x \notin [0, 2\lambda - 1]$.

This last oracle can be simulated by \mathcal{B} with only oracle access to a random PRF $g_k \in G_\lambda$. That this concept is weakly learnable violates the security of the PRF G in the usual way. \square

A.2 Aggregate queries

We turn to a positive result for learning in the AQ model. Our starting point is the intuition that with aggregate queries, it is easy to distinguish a point function from an everywhere-zero function.

Formally, consider the case when $D = \mathbb{Z}_{2^\lambda}$, $R = \{0, 1\}$, $\Gamma = \sum$ is summation modulo 2, and $\mathcal{S}_{[a,b]} = \{[a, b] : a, b \in \mathbb{Z}_{2^\lambda}; a \leq b\}$ the set of intervals on \mathbb{Z}_λ . By AQ-learning with respect to *summation over intervals*, we mean $(\sum, \mathcal{S}_{[a,b]})$ -AQ learning. Let the concept class \mathcal{D}_λ of point functions be defined:

$$\mathcal{D}_\lambda := \{\delta_y : y \in \mathbb{Z}_{2^\lambda}\}$$

where each δ_y is nonzero only at y .

Lemma A.2 (Point functions). *The concept class of point functions \mathcal{D}_λ is efficiently, exactly, and properly $(\sum, \mathcal{S}_{[a,b]})$ -AQ-learnable.*

Proof. Observe that for $\delta_y \in \mathcal{D}_\lambda$ and interval $[a, b] \subseteq \mathbb{Z}_{2^\lambda}$: $AGG_{\sum, \delta_y}([a, b]) = 1 \iff y \in [a, b]$. This allows us to perform binary search over the domain and find y with at most λ queries to the $AGG_{\sum, \delta_y}(\cdot)$ oracle. \square

But if we don't require exact-learning, point functions are trivially learnable with no queries at all; indeed, the hypothesis $h(x) = 0$ agrees with $\delta_y(x)$ at all but a single point! But \mathcal{D}_λ is not *exactly* MQ-learnable. More importantly, for two uniformly selected concepts $\delta_y, \delta_w \leftarrow \mathcal{D}_\lambda$, MQ cannot distinguish membership oracle access to δ_y and δ_w . We will leverage this to construct a much stronger separation.

Let $\mathcal{G}_\lambda = \{g_k : \{0, 1\}^{\lambda-1} \rightarrow \{0, 1\}\}_{k \in \{0, 1\}^{\lambda-1}}$ be a pseudorandom function family with $(\lambda - 1)$ -bit keys k and inputs x .

Functions in our concept class $f_k \in \mathcal{F}_\lambda$ will be indexed by an $(\lambda - 1)$ -bit key, but take inputs from $\{0, 1\}^\lambda$. On half the domain, f_k behaves as the PRF g_k , while on the other half it behaves as the point function δ_k . Letting $x[2 : \lambda] = (x[2], \dots, x[\lambda])$:

$$f_k(x) = \begin{cases} \delta_k(x[2 : \lambda]) & \text{if } x[0] = 0 \\ G_k(x[2 : \lambda]) & \text{if } x[1] = 1 \end{cases}$$

Theorem A.3 (Separating AQ from MQ). *The concept class \mathcal{F} is exactly and (properly) AQ-learnable with respect to summation over intervals. For any polynomials $p_\epsilon(\lambda), p_\delta(\lambda)$, this concept class is hard to (ϵ, δ) -MQ learn for $\epsilon \leq \frac{1}{4} - \frac{1}{p_\epsilon(\lambda)}$ and $1 - \delta \geq \frac{1}{p_\delta(\lambda)}$.*

Note that it while it easy to $(1/4, 1/4)$ -MQ learn \mathcal{C} (for example, outputting the constant 0 function), the theorem above claims that we cannot do appreciably better in ϵ with non-negligible probability $1 - \delta$. This has the flavor of a 'hardness of weakly learning' theorem.

Proof. For $\lambda \in \mathbb{N}$, let $f_k \in \mathcal{F}_\lambda$. The first part of the theorem follows as a corollary to the previous lemma. After exactly learning δ_k by binary search, the function f_k is uniquely specified by k .

For the second part, we reduce to the hardness of MQ learning the pseudorandom function, g_k . Suppose for contradiction that there exists an algorithm \mathcal{A} that, when given access to an oracle $\mathcal{O} = g_k(\cdot)$, with probability at least $\frac{1}{p_\delta(\lambda)}$, outputs hypothesis $h : \{0, 1\}^\lambda \rightarrow \{0, 1\}$ with $\Pr_{x \leftarrow \{0, 1\}^\lambda}[h(x) = f_k(x)] \geq \frac{3}{4} + \frac{1}{p_\epsilon}$. We describe \mathcal{B} – a weak MQ-learning algorithm for the concept $\mathcal{G} = \{g_k\}_{k \in K}$. Given access to oracles $\mathcal{O}_\delta = \delta_k(\cdot)$ and $\mathcal{O}_G = g_k(\cdot)$, \mathcal{B} can exactly simulate oracle access to \mathcal{O} and thus output hypothesis h with the same distribution. But with only $t(\lambda)$ -many

queries for any t , the probability (over the random choice of k) of querying a non-zero point in \mathcal{O}_δ is at most $t(\lambda)/2^{\lambda-1}$; thus, with high probability, all queries to \mathcal{O}_δ will be zero. Therefore it is computationally infeasible to distinguish between the pair of oracles $(\mathcal{O}_\delta, \mathcal{O}_G)$ and $(\mathcal{O}_0, \mathcal{O}_G)$, where \mathcal{O}_0 is the constant zero oracle.

If \mathcal{B} answers \mathcal{A} 's oracle queries with $(\mathcal{O}_0, \mathcal{O}_G)$ instead of $(\mathcal{O}_{\delta_k}, \mathcal{O}_G)$, \mathcal{A} will successfully output h which ϵ' approximates f_k with probability $1 - \delta'$. By the indistinguishability argument, $\epsilon' \geq \epsilon - \text{negl}(\lambda) \geq \epsilon/2$ and $1 - \delta' \geq 1 - \delta - \text{negl}(\lambda) \geq 1 - \delta/2$.

Let $h|_b$ be the restriction of h to the set $\{x : x[1] = b\}$ for $b \in \{0, 1\}$.

$$\begin{aligned} \Pr_x[h(x) \neq f_k(x)] &= \frac{1}{2}(\Pr_x[h|_0(x) = 0] + \Pr_x[h|_1(x) = g_k(x[2:n])]) \geq \frac{3}{4} + \frac{1}{2p_\epsilon} \\ &\implies \Pr_x[h|_1(x) = g_k(x[2:n])] \geq \frac{1}{2} + \frac{1}{p_\epsilon(n)}. \end{aligned}$$

Outputting $h|_1$, \mathcal{B} manages to weakly MQ learn the concept \mathcal{G}_λ . That this concept is weakly learnable violates the security of the PRF G in the usual way. \square

Constrained Key-Homomorphic PRFs from Standard Lattice Assumptions

Or: How to Secretly Embed a Circuit in Your PRF

Zvika Brakerski*

Vinod Vaikuntanathan†

Abstract

Boneh et al. (Crypto 13) and Banerjee and Peikert (Crypto 14) constructed pseudorandom functions (PRFs) from the Learning with Errors (LWE) assumption by embedding combinatorial objects, a path and a tree respectively, in instances of the LWE problem. In this work, we show how to generalize this approach to embed *circuits*, inspired by recent progress in the study of Attribute Based Encryption.

Embedding a universal circuit for some class of functions allows us to produce *constrained keys* for functions in this class, which gives us the first standard-lattice-assumption-based constrained PRF (CPRF) for general bounded-description bounded-depth functions, for arbitrary polynomial bounds on the description size and the depth. (A constrained key w.r.t a circuit C enables one to evaluate the PRF on all x for which $C(x) = 1$, but reveals nothing on the PRF values at other points.) We rely on the LWE assumption and on the one-dimensional SIS (Short Integer Solution) assumption, which are both related to the worst case hardness of general lattice problems. Previous constructions for similar function classes relied on such exotic assumptions as the existence of multilinear maps or secure program obfuscation. The main drawback of our construction is that it does not allow collusion (i.e. to provide more than a single constrained key to an adversary). Similarly to the aforementioned previous works, our PRF family is also *key homomorphic*.

Interestingly, our constrained keys are very short. Their length does not depend directly either on the size of the constraint circuit or on the input length. We are not aware of any prior construction achieving this property, even relying on strong assumptions such as indistinguishability obfuscation.

1 Introduction

A pseudorandom function family (PRF) [GGM86] is a finite set of functions $\{F_s : D \rightarrow R\}_s$, indexed by a seed (or key) s , such that for a random s , F_s is efficiently computable given s , and is computationally indistinguishable from a random function from D to R , given oracle access. Since the introduction of this concept, PRFs have been one of the most fundamental building blocks in cryptography. Many variants of PRFs with additional properties have been introduced and have found a plethora of applications in cryptography. In this work, we will focus on *Constrained PRFs* and *Key-Homomorphic PRFs*.

*Weizmann Institute of Science, Rehovot, Israel. Email: zvika.brakerski@weizmann.ac.il. Supported by ISF grant 468/14, and by an Alon Young Faculty Fellowship.

†MIT, Cambridge, MA, USA. Email: vinodv@csail.mit.edu. Research supported by DARPA Grant number FA8750-11-2-0225, Alfred P. Sloan Research Fellowship, NSF CAREER Award CNS-1350619, NSF Frontier Grant CNS-1414119, Microsoft Faculty Fellowship, and a Steven and Renee Finn Career Development Chair from MIT.

Constrained PRFs. Constrained PRFs (CPRFs) have been introduced simultaneously by Boneh and Waters [BW13], Kiayias et al. [KPTZ13] (as “Delegatable PRFs”) and by Boyle, Goldwasser and Ivan [BGI14] (as “Functional PRFs”). Here an adversary is allowed to ask for a constrained key which should allow it to evaluate the PRF on a subset of the inputs, while revealing nothing about the values at other inputs. It has been shown [BW13, KPTZ13, BGI14] how to construct CPRFs for function classes of the form $x \in [i, j]$ (where the input is interpreted as an integer) based on any one-way function. This in particular allows for the “puncturing” technique of Sahai and Waters [SW14] that found many uses in the obfuscation literature. Further, [BW13] showed how to achieve more complicated function classes such as bit fixing functions and even arbitrary circuits, but those require use of cryptographic multilinear maps. They also introduce a number of applications for such CPRFs, including broadcast encryption schemes and identity based key exchange. Hofheinz et al. [HKKW14] show how to achieve adaptively secure CPRFs from indistinguishability obfuscation using a random oracle.

The original definition of CPRFs requires resilience to arbitrary collusion. Namely, a constrained key for C_1, C_2 should give no more information than a constrained key for $C_1 \vee C_2$ and must not reveal anything about values where $C_1(x) = C_2(x) = \text{false}$. Many of the applications of CPRFs (e.g. for broadcast encryption and identity based key exchange) rely on collusion resilience. Unfortunately, our construction in this work will not allow collusions, and therefore will not be useful for these applications. We hope that future works will be able to leverage our ideas into collusion resilient CPRFs.

Key-Homomorphic PRFs. In key-homomorphic PRFs, there is a group structure associated with the set of keys, and it is required that for any input x and keys s, t , $F_s(x) + F_t(x) = F_{s+t}(x)$. A construction in the random oracle model was given by Naor, Pinkas and Reingold [NPR99], and the first construction in the standard model was given by Boneh et al. [BLMR13] based on the Learning with Errors assumption (LWE), building on a (non key homomorphic) lattice-based PRF of Banerjee, Peikert and Rosen [BPR12]. This was followed by an improved construction by Banerjee and Peikert [BP14] based on quantitatively better lattice assumptions. The LWE based constructions achieved a slightly weaker notion, namely “almost” key-homomorphism, in which $\|(F_s(x) + F_t(x)) - F_{s+t}(x)\|$ is small, for an appropriately defined norm. This notion is sufficient for the known applications. Applications of key-homomorphic PRFs include distributed key-distribution, symmetric proxy re-encryption, updatable encryption and PRFs secure against related-key attacks [NPR99, BLMR13, LMR14].

Our Results. We view the main contribution of this work as showing how to impose *hidden semantics* into the evaluation process of LWE-based PRFs. Namely, we allow multiple computation paths for computing $F_s(x)$, such that we can selectively block some of these paths based on logic described by a circuit. This is done by extending ideas from the ABE literature, and in particular the ABE scheme of Boneh et al. [BGG⁺14] (see more about this connection below).

It is particularly interesting that previous constructions of PRFs [BLMR13, BP14] can be viewed as a special case of our framework, but ones that only allow a single computational path. Our work therefore highlights that the techniques used for constructing PRFs and for constructing ABE are special cases of the same grand schema. This could hopefully lead to new insights and constructions.

We employ our methods towards presenting a family of (single key secure) constrained key-

homomorphic PRFs based on worst-case general lattice assumptions. This is a first step in solving the open problem posed in [BW13] of achieving (collusion resilient) CPRFs from standard assumptions.

Our construction is selectively secure in the constraint query, namely the adversary needs to decide on the constraint before seeing the public parameters, but is adaptive with regards to PRF oracle queries. We achieve the latter without “complexity leveraging”, contrary to [BW13], and thus we do not require sub-exponential hardness assumptions as they do. This is done by employing our technique of embedding semantics into the evaluation process again. In particular, we embed the semantics of an *admissible hash function*, introduced by Boneh and Boyen [BB04] into the PRF, which allows us to handle adaptive queries.

Our proofs rely on two closely related hardness assumptions: The Learning with Errors (LWE) assumption, and the one-dimensional Short Integer Solution (1D-SIS) assumption. Both assumptions can be tied to the worst case hardness of general lattice problems such as GapSVP and SIVP, with similar parameters. LWE is sufficient for proving pseudorandomness in the absence of a constrained key. However, once the adversary is given a constrained key, the situation becomes more delicate. In particular, even showing *correctness* in this setting is not straightforward. (Correctness refers to the property that evaluation using the constrained key and using the actual seed result in the same output.) One can show unconditionally that the value computed using the constrained key is *close* (in norm) to the real value of the function but not that they are always equal. A similar issue comes up in the security proof (since the reduction “fabricates” oracle answers in a similar way to the constrained evaluation). Our solution is to use *computational* arguments. Namely to show that it is computationally intractable, under the 1D-SIS assumption, to come up with an input for which the constrained evaluation errs. Therefore even the correctness of our scheme relies on computational assumptions. We note that similar techniques can be used to strengthen the almost key-homomorphism property into computational key-homomorphism where it is computationally hard to find an input for which key homomorphism does not hold.

The following theorem presents the simplest application of our method, we explain how it can be extended below.

Theorem 1.1. *Let $\mathcal{C}_{\ell,d}$ be the class of size- ℓ depth- d circuits. Then for all polynomials ℓ, d , there exists a $\mathcal{C}_{\ell,d}$ -constrained (almost) key-homomorphic family of PRFs without collusion, based on the (appropriately parameterized) LWE and 1D-SIS assumptions (and hence on the worst-case hardness of appropriately parameterized GapSVP and SIVP problems).*

Interestingly, we can go beyond bounded size circuits. In fact, we can support any function family with bounded length description, so long as there is a universal evaluator of depth d that takes a function description and an input, and executes the function on the input. Namely, consider a sequence of universal circuits $\{\mathcal{U}_k\}_{k \in \mathbb{N}}$, where $\mathcal{U}_k : \{0, 1\}^\ell \times \{0, 1\}^k \rightarrow \{0, 1\}$. This sequence defines a class of functions $\{0, 1\}^* \rightarrow \{0, 1\}$, where each function F in the class is represented by a string $f \in \{0, 1\}^\ell$, and for $x \in \{0, 1\}^k$, it holds that $F(x) = \mathcal{U}_k(f, x)$. We call such a function class ℓ -uniform. We are only able to support \mathcal{U}_k whose depth is bounded by some a-priori polynomial in the security parameter d , however in some cases this is sufficient to support all k 's that are polynomial in the security parameter. The following theorem states our result with regards to such families.

Theorem 1.2. *Let $\mathcal{C}_{\ell,d}$ be a class of ℓ -uniform functions with depth- d evaluator. Then for all polynomials ℓ, d , there exists a $\mathcal{C}_{\ell,d}$ -constrained (almost) key-homomorphic family of PRFs without*

collusion, based on the (appropriately parameterized) LWE and 1D-SIS assumptions (and hence on the worst case hardness of appropriately parameterized GapSVP, SIVP).

Lastly, we show that the bit-length of the constrained keys in our scheme can be reduced to $\text{poly}(\lambda)$ for some *fixed* polynomial. Namely, completely independent of all of the parameters of the scheme. This is done by using an ABE scheme with short secret keys as a black box. In particular we resort to the same scheme, namely the ABE scheme of Boneh et al. [BGG⁺14], which inspired our constrained PRF construction. This is done by encrypting all of the “components” of the constrained key, and providing them in the public parameters of the construction. Then, the actual constrained key is an ABE secret key which only allows to decrypt the relevant components. We note that this short representation for constrained keys is not homomorphic (however the scheme is still almost key homomorphic with respect to the seed). A theorem statement follows.

Theorem 1.3. *There exists a constrained PRF scheme with the same properties as in Theorem 1.2, and under the same hardness assumptions, where the constrained keys are of asymptotic bit-length $\text{poly}(\lambda)$, for an a-priori fixed polynomial.*

See Section 2 for an extended overview of the construction.

Relation to the ABE Construction of Boneh et al. [BGG⁺14]. Our techniques are greatly influenced by the aforementioned LWE-based ABE construction of Boneh et al. [BGG⁺14]. Recall that in ABE, messages are encrypted relative to *attributes* and decryption keys are drawn relative to *functions*. Decryption is possible only if the function f of the decryption key accepts the attribute x of the ciphertext. In order to decrypt a ciphertext, [BGG⁺14] first applies a public procedure that depends on f, x on the ciphertext and then applies the decryption key on the resulting value. Their construction makes sure that for any f , encryptions with regards to all accepting x ’s will derive a decryptable ciphertext (and all non-accepting x ’s cannot be decrypted).

Our constrained key for a circuit C is almost identical to an encryption of 0 with attribute C in [BGG⁺14]. The randomness in the encryption roughly corresponds to the seed of the PRF. An application of the PRF on the constrained key includes applying the public procedure of the ABE on the ciphertext, with respect to the function $f = \mathcal{U}$, the universal circuit for the function class to which C belongs. However, there is the question of how to represent the input: We need to be able to evaluate C on any possible input while preserving security. One of our main technical ideas is in showing that this is possible, and in fact can be achieved regardless of the input length. Combined with the framework from [BGG⁺14], we can guarantee that for all x , regardless which C was used to generate the “ciphertext”, the output of the public procedure will only depend on x and not on C . The basic idea is therefore to use this value as the PRF value. This does not work as is (for example, it does not imply pseudorandomness for non-accepting x ’s) and additional ideas are required.

As mentioned above, the PRFs of [BLMR13, BP14] that seem to stem from different ideas and have quite different proofs than [BGG⁺14] can be shown to be special cases of the above paradigm, except f is taken to be an arbitrary formula (a multiplication tree). For details see Section 2.

The novelty in our approach is to show the extra power that is obtained from generalizing these two approaches. We use the universal circuit as a way to embed an undisclosed computation into an LWE instance, and show how to achieve pseudorandomness using tools such as admissible hash functions (which are also embedded into an LWE instance).

Relation with the Constrained PRF of Hofheinz et al. [HKKW14]. The work of [HKKW14] constructs adaptively secure collusion-resistant CPRFs, namely ones where the challenge x^* needs not be provided ahead of time. Their building blocks are “universal parameters” and *adaptively secure* ABE, which are used as black-box. Note that we achieve adaptive security w.r.t the challenge (but not with respect to the constraint) while relying on techniques which are only known to imply *selectively secure* ABE. Further, whereas [HKKW14] use ABE only to implement access control and therefore need to rely on strong assumptions to implement the PRF so as to interface with the ABE, we use ABE techniques to achieve both pseudorandomness and access control. On the flip side, our construction is not collusion resistant, contrary to [HKKW14].

Open Problems. The main drawback of our CPRF is its vulnerability to collusion, which severely limits its applicability as a building block. It is an open problem to achieve bounded collusion resilience, even for two constrained keys instead of one and even at the cost of increasing the parameters. Any improvement on this front should be very interesting. Another avenue for research is trying to extend the construction so that there is no restriction on the constraint circuit size, similarly to the multilinear map based construction of [BW13]. Finally, it would also be interesting to apply this methodology of imposing semantics on a cryptographic computation to other primitives in order to allow more fine-grained access control.

2 Overview of Our Construction

We recall that the LWE assumption asserts that for a uniform vector \mathbf{s} and a matrix \mathbf{A} of appropriate dimensions (over \mathbb{Z}_q for an appropriate q), it holds that $(\mathbf{A}, \mathbf{s}^T \mathbf{A} + \mathbf{e}^T)$, is indistinguishable from uniform, where \mathbf{e} is taken from an appropriate distribution over *low norm* vectors and referred to as the *noise vector*. In this outline we will ignore the generation of \mathbf{e}^T and its evolution during computation process, and just denote it by *noise* (but of course care will need to be taken in the formal arguments).

The PRF of Banerjee and Peikert [BP14]. A high-level methodology for constructing PRFs, taken by [BLMR13, BP14] and also in this work, is to take \mathbf{s} as the seed, and to generate for each PRF input x , an LWE matrix \mathbf{A}_x such that the values $\mathbf{s}^T \mathbf{A}_x + \text{noise}$ for the different inputs x are jointly indistinguishable from uniform. Note that almost key homomorphism follows naturally for any implementation of this template, up to the accumulation of noise. The noise issue is handled by taking the PRF value to be a properly scaled down and rounded version of the above, so that the effect of the noise is minimal (and its norm can be bounded below 1). This property is also inherited by our scheme.

As a starting point for deriving our construction, let us revisit the key-homomorphic PRF construction of [BP14]. Their PRF family was associated with a combinatorial object – a binary tree. Each node v of the tree was associated with an LWE matrix \mathbf{A}_v , where the PRF input x determined the matrices for the leaves, and matrices for internal nodes are derived as follows. Given a node v whose children are associated with $\mathbf{A}_l, \mathbf{A}_r$, they define $\mathbf{A}_v = \mathbf{A}_l \cdot \mathbf{G}^{-1}(\mathbf{A}_r)$. In this notation, $\mathbf{G}^{-1}(\cdot)$ is the binary decomposition operator, which breaks each entry in the matrix into the bit vector of length $\log(q)$ of its binary representation. Note that $\mathbf{G}^{-1}(\cdot)$ will always have small norm, and that the inverse operator \mathbf{G} , representing binary composition, is linear so it can be represented by a matrix. Thus for all \mathbf{A} it holds that $\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$.

Going back to the PRF of [BP14], the derivation procedure described above allows to associate a matrix with the root of the tree, which depends only on the input x (and on the topology of the tree which is fixed). We will use the root's matrix as our \mathbf{A}_x . The proof hinges on the invariant that LWE instances will be multiplied on the right only by low-norm matrices (of the form $\mathbf{G}^{-1}(\cdot)$), and therefore $\mathbf{s}^T \mathbf{A}_l \mathbf{G}^{-1}(\mathbf{A}_r) + \text{noise} \approx (\mathbf{s}^T \mathbf{A}_l + \text{noise}) \mathbf{G}^{-1}(\mathbf{A}_r)$, which allows to replace $(\mathbf{s}^T \mathbf{A}_l + \text{noise})$ with a new uniform vector and propagate to the right.

From Embedded Trees to Embedded Circuits. We show that the operation $\mathbf{A}_v = \mathbf{A}_l \cdot \mathbf{G}^{-1}(\mathbf{A}_r)$ is in fact a special case of a more general operation, inspired by the recent Attribute Based Encryption (ABE) construction of Boneh et al. [BGG⁺14]. We will associate a matrix \mathbf{A}_v as well as a binary value x_v with each node, and pay special attention to the matrix $(\mathbf{A}_v - x_v \mathbf{G})$. In particular, considering a node v with children l, r , it holds that

$$(\mathbf{A}_l - x_l \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{A}_r) + (\mathbf{A}_r - x_r \mathbf{G}) \cdot x_l = \mathbf{A}_l \mathbf{G}^{-1}(\mathbf{A}_r) - x_r x_l \mathbf{G}.$$

This generalization associates the semantics of the multiplication operation with the syntactic definition $\mathbf{A}_v = \mathbf{A}_l \mathbf{G}^{-1}(\mathbf{A}_r)$, and it also maintains the invariant that the matrices $(\mathbf{A}_l - x_l \mathbf{G})$ and $(\mathbf{A}_r - x_r \mathbf{G})$ are only multiplied on the right by low norm elements, so that

$$\begin{aligned} \mathbf{s}^T \left((\mathbf{A}_l - x_l \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{A}_r) + (\mathbf{A}_r - x_r \mathbf{G}) \cdot x_l \right) + \text{noise} \approx \\ \left(\mathbf{s}^T (\mathbf{A}_l - x_l \mathbf{G}) + \text{noise} \right) \cdot \mathbf{G}^{-1}(\mathbf{A}_r) + \left(\mathbf{s}^T (\mathbf{A}_r - x_r \mathbf{G}) + \text{noise} \right) \cdot x_l, \end{aligned}$$

which will play an important role in the security proof. Put explicitly, if the evaluator holds $\mathbf{s}^T (\mathbf{A}_l - x_l \mathbf{G}) + \text{noise}$ and $\mathbf{s}^T (\mathbf{A}_r - x_r \mathbf{G}) + \text{noise}$, then it can compute $\mathbf{s}^T (\mathbf{A}_v - x_l \cdot x_r \mathbf{G}) + \text{noise}$ (and we will obviously define $x_v = x_l \cdot x_r$).

This semantic relation can be extended beyond multiplication gates, and in particular NAND gates can be supported in a fairly similar manner. Furthermore, there is no need to stick to tree structure and one can support arbitrary DAGs, which naturally correspond to circuits. Extending the above postulate, if our DAG corresponds to a circuit C , then having $\mathbf{s}^T (\mathbf{A}_i - x_i \mathbf{G}) + \text{noise}$, for all leaves (= inputs), allows to compute $\mathbf{s}^T (\mathbf{A}_x - C(x) \mathbf{G}) + \text{noise}$. Recalling that the value of the PRF on input x is $\mathbf{s}^T \mathbf{A}_x + \text{noise}$, the aforementioned information allows us to evaluate the PRF at points where $C(x) = 0$. It can also be shown that it is computationally hard to compute the value at points where $C(x) = 1$. We note that this process is practically identical to the public part of the decryption procedure in the [BGG⁺14] ABE (as we explained in Section 1). We also note that since [BP14] were trying to minimize the complexity of evaluating their PRF, it made no sense in their construction to consider DAGs which only increase the complexity. However, as we show here, there are benefits to embedding a computational process in the PRF evaluation.

Utilizing the Universal Circuit. The tools we describe so far indeed seem to get us closer to our goal of producing constrained keys, but we are still not quite there. What we showed is that for any circuit C , we can devise a PRF with a constrained key for C . Note that we use the negated definition to the one we used before, and allow to evaluate when $C(x) = 0$ and not when $C(x) = 1$. This will be our convention throughout this overview.

In order to reverse the order of quantifiers, we take C to be the universal circuit $\mathcal{U}(F, x)$, and the constrained keys will be of the form $\mathbf{s}^T (\mathbf{A}_i - f_i \mathbf{G}) + \text{noise}$, where the f_i is the i th bit of the description

of the constraint F , as well as values for the x wires, which will be of the form $\mathbf{s}^T(\hat{\mathbf{A}}_b - b\mathbf{G}) + \text{noise}$, for both $b \in \{0, 1\}$. These values will allow us to execute F on *any* input x . Note that we can use the same matrices $\hat{\mathbf{A}}_0, \hat{\mathbf{A}}_1$ for all input wires, hence we don't need to commit to the input size when we provide the constrained key.¹ From this description it is obvious why our construction is not collusion resistant: Given two constrained keys for two non identical functions, there exists an i such that the adversary gets both $\mathbf{s}^T\mathbf{A} + \text{noise}$ and $\mathbf{s}^T(\mathbf{A}_i - \mathbf{G}) + \text{noise}$. Recovering \mathbf{s}^T from these values is straightforward and hence all security is lost. Note that for the input values, unlike the function description, we use two different matrices for 0 and 1: $\hat{\mathbf{A}}_0, \hat{\mathbf{A}}_1$, so a similar problem does not occur.

The Problem with Correctness, and a Computational Solution. We introduced two ways to compute the value of the PRF at x : One is to compute \mathbf{A}_x and use the seed \mathbf{s}^T to compute $\mathbf{s}^T\mathbf{A}_x + \text{noise}$, and the other is to use the constrained key to obtain $\mathbf{s}^T(\mathbf{A}_x - F(x)\mathbf{G}) + \text{noise}$, which for $F(x) = 0$ gives $\mathbf{s}^T\mathbf{A}_x + \text{noise}$. The problem is that the *noise value* in these two methods could differ. It is possible to make the difference small by scaling down and rounding, but this is not going to suffice for our purposes (mostly because a similar problem comes up in the security proof). We solve this issue using the 1D-SIS assumption as follows. We first note that the evaluation using the constrained key is essentially evaluation of a linear function with small coefficients on the vectors constituting the constrained key (essentially they get multiplied by bits and by low norm matrices $\mathbf{G}^{-1}(\cdot)$). Secondly, the only way for the two computation paths to not agree is if the value $\mathbf{s}^T\mathbf{A}_x$ is very close to an integer multiple of a number p (which is part of the PRF description). Finally, we notice that by LWE, the vectors in the constrained key are indistinguishable from uniform and independent. Thus, if we encounter such x for which correctness does not work, we can also find a short linear combination of random elements whose scaled down rounded value is close to an integer. In other words, given a uniform vector \mathbf{v} in \mathbb{Z}_q , we can find \mathbf{z} such that $\lfloor \langle \mathbf{v}, \mathbf{z} \rangle / p \rfloor$ is “close” to an integer. This is similar to solving a one-dimensional instance of the SIS problem, i.e. $\langle \mathbf{v}, \mathbf{z} \rangle = 0 \pmod{p}$. Indeed, one can show that the 1D-SIS problem is as hard as standard worst-case hard lattice problems via a reduction from [Reg04].

Pseudorandomness and Adaptive Security. Given a constrained key for F , one can compute $\mathbf{s}^T(\mathbf{A}_x - F(x)\mathbf{G}) + \text{noise}$, and indeed if $F(x) = 1$ it is hard to compute $\text{PRF}_s(x) = \mathbf{s}^T\mathbf{A}_x + \text{noise}$. However, we want to argue that this value is *pseudorandom* and furthermore that it remains pseudorandom after adaptive queries to the PRF. Namely, after the adversary sees as many values of the form $\text{PRF}_s(x) = \mathbf{s}^T\mathbf{A}_x + \text{noise}$ as it wishes.

To achieve these goals, we add another feature to the PRF. We consider a new independent LWE matrix \mathbf{D} , and define $\text{PRF}_s(x) = \mathbf{s}^T\mathbf{A}_x \cdot \mathbf{G}^{-1}(\mathbf{D}) + \text{noise}$. First of all, we note that given the constrained key, we can still compute the PRF for values where $C(x) = 0$, by first computing $(\mathbf{s}^T\mathbf{A}_x + \text{noise})$ as before, and then multiplying by $\mathbf{G}^{-1}(\mathbf{D})$, which has low norm. However, in general we have

$$\text{PRF}_s(x) \approx \left(\mathbf{s}^T(\mathbf{A}_x - F(x)\mathbf{G}) + \text{noise} \right) \cdot \mathbf{G}^{-1}(\mathbf{D}) + F(x) \left(\mathbf{s}^T\mathbf{D} + \text{noise} \right),$$

and it can be shown that for $F(x) = 1$, the second term randomizes the expression, by the LWE assumption.

¹Recall that in [BLMR13, BP14] there are only two matrices altogether. This is sufficient here for the input wires for the same reason, but we need additional matrices to encode the constraint description.

This handles pseudorandomness for a single query, but not for the case of adaptive queries (since we can only use the pseudorandomness of $(\mathbf{s}^T \mathbf{D} + \text{noise})$ once). To handle adaptive queries we embed semantics into the matrix \mathbf{D} itself. Namely, $\mathbf{D} = \mathbf{D}_x$ will be derived by an application of the universal circuit to the input x and *an admissible hash function* h . Admissible hash functions, introduced by Boneh and Boyen [BB04], allow (at a very high level) to partition the input space such that with noticeable probability all of the adaptive queries have value $h(x) = 0$, but the challenge query will have $h(x) = 1$. This means that in the proof of security, we can hold a constrained key for h , which will allow us to compute $(\mathbf{s}^T \mathbf{D}_x + \text{noise})$, for all the queries of the adversary, but leave the challenge query unpredictable (to make it pseudorandom, we will multiply in the end by another final \mathbf{D}'). This concludes the security argument for adaptive queries.

Key-Homomorphism. As we mention above, key-homomorphism follows since we use the template $\text{PRF}_{\mathbf{s}}(x) = \mathbf{s}^T \mathbf{A}_x + \text{noise}$. We note that the existence of noise means that homomorphism may not be accurate and with some low probability $(\text{PRF}_{\mathbf{s}}(x) + \text{PRF}_{\mathbf{s}'}(x))$ will only be close to $\text{PRF}_{\mathbf{s}+\mathbf{s}'}(x)$ and not identical. However this property is sufficient for many applications.

We point out that our constrained keys are a collection elements of the form $(\mathbf{s}^T \mathbf{A}_i + \text{noise})$, and therefore the scheme is also homomorphic with respect to constrained keys, i.e. constrained keys for the same F w.r.t different keys \mathbf{s}, \mathbf{s}' can be added to obtain a constrained key w.r.t $\mathbf{s} + \mathbf{s}'$.

Reducing the Constrained Key Size. From the above, it follows that the constrained key contains $\ell + 2$ vectors, where ℓ is the bit length of a description of F relative to the universal circuit for the function class. Note that this does not depend directly on the input size to the function. However, indirectly the depth of the universal circuit affects the modulus q that needs to be used.

We show that we can remove the dependence on ℓ altogether using an ABE scheme with short secret keys, such as that of [BGG⁺14]. To do this, we notice that for each constraint function F , the adversary gets either $\mathbf{s}^T \mathbf{A}_i + \text{noise}$ or $\mathbf{s}^T (\mathbf{A}_i - \mathbf{G}) + \text{noise}$, according to the value of the bit f_i . We can prepare for both options by encrypting both vectors using the ABE, each with its own attribute $(i, 0)$ and $(i, 1)$ respectively. All of these encryptions, for all i , will be placed in the public parameters. Then in order to provide a constrained key, we will provide an ABE secret key for the function that takes (i, b) and returns 0 if and only if $f_i = b$. Given this key, the user can decrypt exactly those vectors that constitute its constrained key. Note that this function can be computed by a depth $O(\log(\ell)) = O(\log(\lambda))$ circuit, and thus the size of the secret key can be made asymptotically independent of all parameters except λ , e.g. by setting the parameters to support depth $\log^2(\lambda)$ circuits.

3 Preliminaries

We first recall some background. For an integer modulus q , let $\mathbb{Z}_q = \mathbb{Z}/q\mathbb{Z}$ denote the ring of integers modulo q . For an integer $p \leq q$, we define the modular “rounding” function

$$\lfloor \cdot \rfloor_p : \mathbb{Z}_q \rightarrow \mathbb{Z}_p \text{ that maps } x \rightarrow \lfloor (p/q) \cdot x \rfloor$$

and extend it coordinate-wise to matrices and vectors over \mathbb{Z}_q . We denote the elements of the standard basis by $\mathbf{u}_1, \mathbf{u}_2, \dots$, where the dimension will be clear from the context.

We denote distributions (or random variables) that are computationally indistinguishable by $X \stackrel{c}{\approx} Y$. This refers to the standard notion of negligible distinguishing gap for any polynomial

time distinguisher. Our reductions preserve the uniformity of the adversary so by assuming the hardness of our assumption for uniform adversary we get security for our construction against uniform adversaries, and likewise for non-uniform assumptions and adversaries.

The Gadget Matrix. Let $\ell = \lceil \log q \rceil$ and define the “gadget matrix” $\mathbf{G} = \mathbf{g} \otimes \mathbf{I}_n \in \mathbb{Z}_q^{n \times n\ell}$ where

$$\mathbf{g} = (1, 2, 4, \dots, 2^{\ell-1}) \in \mathbb{Z}_q^\ell$$

We will also refer to this gadget matrix as the “powers-of-two” matrix. We define the inverse function $\mathbf{G}^{-1} : \mathbb{Z}_q^{n \times m} \rightarrow \{0, 1\}^{n\ell \times m}$ which expands each entry $a \in \mathbb{Z}_q$ of the input matrix into a column of size ℓ consisting of the bit decomposition of a . We have the property that for any matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$,

$$\mathbf{G} \cdot \mathbf{G}^{-1}(\mathbf{A}) = \mathbf{A}$$

Norms for Vectors and Matrices. We will always use the infinity norm for vectors and matrices. Namely for a vector \mathbf{x} , the norm $\|\mathbf{x}\|$ is the maximal absolute value of an element in \mathbf{x} . Similarly, for a matrix \mathbf{A} , $\|\mathbf{A}\|$ is the maximal absolute value of any of its entries. If \mathbf{x} is n -dimensional and \mathbf{A} is $n \times m$, then $\|\mathbf{x}^T \mathbf{A}\| \leq n \cdot \|\mathbf{x}\| \cdot \|\mathbf{A}\|$. We remark that L_1 or L_2 norms can also be used and even achieve somewhat tighter parameters, but the proofs become more complicated.

3.1 Constrained Pseudorandom Function: Definition

In a constrained PRF family [BW13, BGI14, KPTZ13], one can compute a constrained PRF key K_C corresponding to any Boolean circuit C . Given K_C , anyone can compute the PRF on inputs x such that $C(x) = 0$. Furthermore, K_C does not reveal any information about the PRF values at the other locations. Below we recall their definition, as given by [BW13].

Syntax A *constrained* pseudo-random function (PRF) family is defined by a tuple of algorithms (KeyGen, Eval, Constrain, ConstrainEval) where:

- **Key Generation** $\text{KeyGen}(1^\lambda, 1^{k_{\text{in}}}, 1^{k_{\text{out}}})$ is a PPT algorithm that takes as input the security parameter λ , an input length k_{in} and an output length k_{out} , and outputs a PRF key K ;
- **Evaluation** $\text{Eval}(K, x)$ is a deterministic algorithm that takes as input a key K , a string $x \in \{0, 1\}^{k_{\text{in}}}$ and outputs $y \in \{0, 1\}^{k_{\text{out}}}$;
- **Constrained Key Generation** $\text{Constrain}(K, C)$ is a PPT algorithm that takes as input a PRF key K , a circuit $C : \{0, 1\}^{k_{\text{in}}} \rightarrow \{0, 1\}$ and outputs a constrained key K_C ;
- **Constrained Evaluation** $\text{ConstrainEval}(K_C, x)$ is a deterministic algorithm that takes as input a constrained key K_C and a string $x \in \{0, 1\}^{k_{\text{in}}}$ and outputs either a string $y \in \{0, 1\}^{k_{\text{out}}}$ or \perp .

We define the notion of (*single key*) *selective-function* security for constrained PRFs.

Definition 3.1. A family of PRFs (KeyGen, Eval, Constrain, ConstrainEval) is a single-key selective-function constrained PRF (henceforth, referred to simply as *constrained PRF*) if it satisfies the following properties:

- **Functionality computationally preserved under constraining.** For every PPT adversary (A_0, A_1) , consider an experiment where we choose $K \leftarrow \text{KeyGen}(1^\lambda, 1^{k_{\text{in}}}, 1^{k_{\text{out}}})$, $(C, \sigma_0) \leftarrow A_0(1^\lambda)$, and $K_C \leftarrow \text{Constrain}(K, C)$. Then:

$$\Pr \left[x^* \leftarrow A_1^{\text{Eval}(K, \cdot)}(1^\lambda, K_C, \sigma_0); \quad : \quad C(x^*) = 0 \wedge \text{Eval}(K, x^*) \neq \text{ConstrainEval}(K_C, x^*) \right]$$

is negligible in the security parameter, where C, K, K_C are selected as described above.

In words, it is computationally hard to find an x^* such that $C(x^*) = 0$, and yet the result of the constrained evaluation differs from the actual PRF evaluation.

- **Pseudorandom at constrained points.** For every PPT adversary (A_0, A_1, A_2) , consider an experiment where $K \leftarrow \text{KeyGen}(1^\lambda, 1^{k_{\text{in}}}, 1^{k_{\text{out}}})$, $(C, \sigma_0) \leftarrow A_0(1^\lambda)$, and $K_C \leftarrow \text{Constrain}(K, C)$. Then:

$$\Pr \left[\begin{array}{l} b \leftarrow \{0, 1\}; \\ (x^*, \sigma_1) \leftarrow A_1^{\text{Eval}(K, \cdot)}(1^\lambda, K_C, \sigma_0); \\ \text{If } b = 0, y^* = \text{Eval}(K, x^*), \\ \text{Else } y^* \leftarrow \{0, 1\}^{k_{\text{out}}} \end{array} : \begin{array}{l} C(x^*) = 1 \wedge \\ A_2(1^\lambda, y^*, \sigma_1) = b \end{array} \right] \leq \frac{1}{2} + \text{negl}(\lambda)$$

The correctness and security properties could potentially be combined into one game, but we choose to present them as two distinct properties for the sake of clarity.

3.2 Learning with Errors

The Learning with Errors (LWE) problem was introduced by Regev [Reg05] as a generalization of “learning parity with noise” [BFKL93, Ale03]. We now define the decisional version of LWE. (Unless otherwise stated, we will treat all vectors as column vectors in this paper).

Definition 3.2 (Decisional LWE (DLWE) [Reg05]). Let λ be the security parameter, $n = n(\lambda)$, $m = m(\lambda)$, and $q = q(\lambda)$ be integers and $\chi = \chi(\lambda)$ be a probability distribution over \mathbb{Z} . The $\text{DLWE}_{n,q,\chi}$ problem states that for all $m = \text{poly}(n)$, letting $\mathbf{A} \leftarrow \mathbb{Z}_q^{n \times m}$, $\mathbf{s} \leftarrow \mathbb{Z}_q^n$, $\mathbf{e} \leftarrow \chi^m$, and $\mathbf{u} \leftarrow \mathbb{Z}_q^m$, the following distributions are computationally indistinguishable:

$$(\mathbf{A}, \mathbf{s}^T \mathbf{A} + \mathbf{e}^T) \stackrel{c}{\approx} (\mathbf{A}, \mathbf{u}^T)$$

There are known quantum (Regev [Reg05]) and classical (Peikert [Pei09]) reductions between $\text{DLWE}_{n,q,\chi}$ and approximating short vector problems in lattices. Specifically, these reductions take χ to be a discrete Gaussian distribution $D_{\mathbb{Z},\alpha q}$ for some $\alpha < 1$. We write $\text{DLWE}_{n,q,\alpha}$ to indicate this instantiation. We now state a corollary of the results of [Reg05, Pei09, MM11, MP12]. These results also extend to additional forms of q (see [MM11, MP12]).

Corollary 3.1 ([Reg05, Pei09, MM11, MP12]). Let $q = q(n) \in \mathbb{N}$ be either a prime power $q = p^r$, or a product of co-prime numbers $q = \prod q_i$ such that for all i , $q_i = \text{poly}(n)$, and let $\alpha \geq \sqrt{n}/q$. If there is an efficient algorithm that solves the (average-case) $\text{DLWE}_{n,q,\alpha}$ problem, then:

- There is an efficient quantum algorithm that solves $\text{GapSVP}_{\tilde{O}(n/\alpha)}$ (and $\text{SIVP}_{\tilde{O}(n/\alpha)}$) on any n -dimensional lattice.

- If in addition $q \geq \tilde{O}(2^{n/2})$, there is an efficient classical algorithm for $\text{GapSVP}_{\tilde{O}(n/\alpha)}$ on any n -dimensional lattice.

Recall that GapSVP_γ is the (promise) problem of distinguishing, given a basis for a lattice and a parameter d , between the case where the lattice has a vector shorter than d , and the case where the lattice doesn't have any vector shorter than $\gamma \cdot d$. SIVP is the search problem of finding a set of “short” vectors. The best known algorithms for GapSVP_γ ([Sch87]) require at least $2^{\tilde{\Omega}(n/\log \gamma)}$ time. We refer the reader to [Reg05, Pei09] for more information.

In this work, we will only consider the case where $q \leq 2^n$. Furthermore, the underlying security parameter λ is assumed to be polynomially related to the dimension n .

3.3 One-Dimensional Short Integer Solution (SIS) and Variants

We present a special case of the well known Short Integer Solution (SIS) problem [Ajt96].

Definition 3.3. *The One-Dimensional Short Integer Solution problem, denoted $1\text{D-SIS}_{q,m,t}$, is the following problem. Given a uniformly distributed vector $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^m$, find $\mathbf{z} \in \mathbb{Z}^m$ such that $\|\mathbf{z}\| \leq t$ and also $\langle \mathbf{v}, \mathbf{z} \rangle \in [-t, t] + q\mathbb{Z}$.*

For appropriately chosen moduli q , the $1\text{D-SIS}_{q,m,t}$ problem is as hard as worst-case lattice problems. This follows from the techniques in the classical worst-case to average-case reduction of Ajtai [Ajt96]. We state below the version due to Regev [Reg04].

Corollary 3.2 (Section 4 in [Reg04] and Proposition 4.7 in [GPV07]). *Let $n \in \mathbb{N}$ and $q = \prod_{i \in n} p_i$, where all $p_1 < p_2 < \dots < p_n$ are co-prime. Let $m \geq c \cdot n \log q$ (for some universal constant c). Assuming that $p_1 \geq t \cdot \omega(\sqrt{mn} \log n)$, the one-dimensional SIS problem $1\text{D-SIS}_{q,m,t}$ is at least as hard as $\text{SIVP}_{t \cdot \tilde{O}(\sqrt{mn})}$ and $\text{GapSVP}_{t \cdot \tilde{O}(\sqrt{mn})}$.*

Proof. The hardness of a closely related problem is established by combining the techniques in [Reg04, Section 4] and [GPV07, Proposition 4.7]: Given $\mathbf{a} \xleftarrow{\$} \mathbb{Z}_q^{m+1}$, find \mathbf{y} with $\|\mathbf{y}\| \leq t$ such that $\langle \mathbf{a}, \mathbf{y} \rangle = 0 \pmod{q}$.

We now show how to convert an instance for this problem into an instance of 1D-SIS . Given an instance $\mathbf{a} \in \mathbb{Z}_q^{m+1}$, we consider the first component a_1 . If this element is not a unit (i.e. invertible) in \mathbb{Z}_q , then the reduction aborts. Otherwise it defines $\mathbf{v} = a_1^{-1} \cdot [a_2, \dots, a_{m+1}]$. Given a solution \mathbf{z} for 1D-SIS on input \mathbf{v} , we define \mathbf{y} by letting $\mathbf{y} = [-\langle \mathbf{v}, \mathbf{z} \rangle, x_1, \dots, x_m]$. It is easy to verify that $\langle \mathbf{a}, \mathbf{y} \rangle = a_1 \cdot (-\langle \mathbf{v}, \mathbf{z} \rangle + \langle \mathbf{v}, \mathbf{z} \rangle) = 0 \pmod{q}$. Further, by definition, $\|\mathbf{y}\| \leq t$. \square

Next, we define a related problem which will be useful for our reductions.

Definition 3.4. *Let $q = p \cdot \prod_{i \in n} p_i$, where all $p_1 < p_2 < \dots < p_n$ are all co-prime and co-prime with p as well. Further let $m \in \mathbb{N}$. The $1\text{D-SIS-R}_{q,p,t,m}$ problem is the following: Given $\mathbf{v} \xleftarrow{\$} \mathbb{Z}_q^m$, find $\mathbf{z} \in \mathbb{Z}^m$ with $\|\mathbf{z}\| \leq t$ such that $\langle \mathbf{v}, \mathbf{z} \rangle \in [-t, t] + (q/p)\mathbb{Z}$.*

The following corollary establishes the hardness of 1D-SIS-R based on 1D-SIS .

Corollary 3.3. *Let q, p, t, m be as in Definition 3.4. Then $1\text{D-SIS-R}_{q,p,t,m}$ is at least as hard as $1\text{D-SIS}_{q/p,t,m}$.*

Proof. The reduction works in the obvious way: Given an input $\mathbf{v} \in \mathbb{Z}_{q/p}^m$ for 1D-SIS $_{q/p,t,m}$, we embed \mathbf{v} in $\mathbf{v}' \in \mathbb{Z}_q^m$, using CRT representation. Namely $\mathbf{v}' = \mathbf{v} \pmod{q/p}$ and $\mathbf{v}' = \mathbf{r} \pmod{p}$, where $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_p^m$. Then given a solution \mathbf{z} for 1D-SIS- $R_{q,p,t,m}$ with input \mathbf{v}' , we claim that \mathbf{z} is also a solution for 1D-SIS $_{q/p,t,m}$ with input \mathbf{v} . This follows since by definition $\|\mathbf{z}\| \leq t$, and since $\langle \mathbf{v}, \mathbf{z} \rangle \equiv \langle \mathbf{v}', \mathbf{z} \rangle \pmod{q/p}$. \square

3.4 Admissible Hash Functions

The concept of admissible hash functions was defined by Boneh and Boyen [BB04] to convert selectively secure identity based encryption (IBE) schemes into fully secure ones. In this paper, we use admissible hash functions for our PRF construction. Our definition of admissible hash functions below will follow that of Cash, Hofheinz, Kiltz and Peikert [CHKP12] with minor changes (in particular, note that we do not require that the bad set is efficiently recognizable).

Definition 3.5 ([BB04, CHKP12]). *Let $\mathcal{H} = \{\mathcal{H}_\lambda\}_\lambda$ be a family of hash functions such that $\mathcal{H}_\lambda \subseteq (\{0,1\}^* \rightarrow \{0,1\}^\ell)$ for some $\ell = \ell(\lambda)$. We say that \mathcal{H} is a family of admissible hash functions if for every $H \in \mathcal{H}$ there exists a set bad_H of “bad string-tuples” such that the following two properties hold:*

1. *For every PPT algorithm \mathcal{A} , there is a negligible function ν such that*

$$\Pr[(x^{(0)}, \dots, x^{(t)}) \in \text{bad}_H \mid H \leftarrow \mathcal{H}_\lambda, (x^{(0)}, \dots, x^{(t)}) \leftarrow \mathcal{A}(1^\lambda, H)] \leq \nu(\lambda)$$

where the probability is over the choice of $H \leftarrow \mathcal{H}_\lambda$ and the coins of \mathcal{A} .

2. *Let $\mathcal{L} = \{0,1\}^{2\ell}$, and for all $L \in \mathcal{L}$ define $\Pi_L : \{0,1\}^\ell \rightarrow \{0,1\}$ to be the string comparison with wildcards function. Namely, write L as a pair of strings $(\alpha, \beta) \in \{0,1\}^\ell$, and define*

$$\Pi_{L=(\alpha,\beta)}(w) = 1 \Leftrightarrow \forall i \in [\ell] \ ((\alpha_i = 0) \vee (\beta_i = w_i)) \ .$$

Intuitively, Π is a string comparison function with wildcards. It compares w and β only at those points where $\alpha_i = 1$. Note that this representation is somewhat redundant but it will be useful for our application.

Then, we require that for every polynomial $t = t(\lambda)$ there exists a noticeable function $\Delta_t(\lambda)$ and an efficiently sampleable distribution \mathcal{L}_t over \mathcal{L} such that for every $H \in \mathcal{H}_\lambda$ and sequences $(x^{(0)}, \dots, x^{(t)}) \notin \text{bad}_H$ with $x^{(0)} \notin \{x^{(1)}, \dots, x^{(t)}\}$, we have:

$$\Pr_{L \leftarrow \mathcal{L}_t} [\Pi_L(H(x^{(0)})) \wedge \overline{\Pi_L(H(x^{(1)}))} \wedge \dots \wedge \overline{\Pi_L(H(x^{(t)}))}] \geq \Delta_t(\lambda)$$

It has been shown by [BB04] that a family of admissible hash functions can be constructed based on any collision resistant hash function. In particular one can instantiate it based on the SIS problem (for virtually any parameter setting for which the problem is hard), which is at least as hard as LWE. Therefore throughout this manuscript we assume the existence of an LWE-based family of admissible hash functions, which will not add an additional assumption to our construction.

3.5 Attribute-Based Encryption

We define (leveled) attribute-based encryption, following [GPSW06, GVW13]. An attribute-based encryption scheme for a class of predicate circuits \mathcal{C} (namely, circuits with a single bit output) consists of four algorithms ($\mathcal{ABE}.\text{Setup}$, $\mathcal{ABE}.\text{KeyGen}$, $\mathcal{ABE}.\text{Enc}$, $\mathcal{ABE}.\text{Dec}$).

$\mathcal{ABE}.\text{Setup}(1^\lambda, 1^\ell, 1^d) \rightarrow (\text{pp}, \text{msk})$: The setup algorithm gets as input the security parameter λ , the length ℓ of the attributes and the maximum depth of the predicate circuits d , and outputs the public parameter (pp, mpk) , and the master key msk . All the other algorithms get pp as part of their input.

$\mathcal{ABE}.\text{KeyGen}(\text{msk}, C) \rightarrow \text{sk}_C$: The key generation algorithm gets as input msk and a predicate specified by $C \in \mathcal{C}$ (of depth at most d). It outputs a secret key (C, sk_C) .

$\mathcal{ABE}.\text{Enc}(\text{pp}, \mathbf{x}, m) \rightarrow \text{ct}$: The encryption algorithm gets as input mpk , attributes $\mathbf{x} \in \{0, 1\}^\ell$ and a message $m \in \mathcal{M}$. It outputs a ciphertext (\mathbf{x}, ct) .

$\mathcal{ABE}.\text{Dec}((C, \text{sk}_C), (\mathbf{x}, \text{ct})) \rightarrow m$: The decryption algorithm gets as input a circuit C and the associated secret key sk_C , attributes \mathbf{x} and an associated ciphertext ct , and outputs either \perp or a message $m \in \mathcal{M}$.

Correctness. We require that for all ℓ, d , all (\mathbf{x}, C) such that $\mathbf{x} \in \{0, 1\}^\ell$, C has depth at most d and $C(\mathbf{x}) = 1$, for all $(\text{pp}, \text{msk}) \leftarrow \mathcal{ABE}.\text{Setup}(1^\lambda, 1^\ell, 1^d)$, all $\text{sk}_C \leftarrow \mathcal{ABE}.\text{KeyGen}(\text{msk}, C)$, all $\text{ct} \leftarrow \mathcal{ABE}.\text{Enc}(\text{pp}, \mathbf{x}, m)$, and all $m \in \mathcal{M}$,

$$\text{Dec}((C, \text{sk}_C), (\mathbf{x}, \text{ct})) = m) .$$

Security Definition. We define selective security of ABE, which is sufficient for our purposes. We allow the adversary to make multiple challenge message queries, which is equivalent to the single query case but will be easier for us to work with.

Definition 3.6. For a stateful adversary \mathcal{A} , we define the advantage function $\text{Adv}_{\mathcal{A}}^{\text{ABE}}$ to be

$$\Pr \left[\begin{array}{l} b \xleftarrow{\$} \{0, 1\}; \\ \mathbf{x}_1, \dots, \mathbf{x}_Q \leftarrow \mathcal{A}(1^\lambda, 1^\ell, 1^d); \\ (\text{pp}, \text{msk}) \leftarrow \mathcal{ABE}.\text{Setup}(1^\lambda, 1^\ell, 1^d); \\ \{(m_{0,i}, m_{1,i})\}_{i \in [Q]} \leftarrow \mathcal{A}^{\mathcal{ABE}.\text{KeyGen}(\text{msk}, \cdot)}(\text{pp}), \forall i. |m_{0,i}| = |m_{1,i}|; \\ \text{ct}_i \leftarrow \mathcal{ABE}.\text{Enc}(\text{pp}, \mathbf{x}_i, m_{b,i}); \\ b' \leftarrow \mathcal{A}^{\mathcal{ABE}.\text{KeyGen}(\text{msk}, \cdot)}(\text{ct}_1, \dots, \text{ct}_Q) \end{array} \right] - \frac{1}{2}$$

with the restriction that all queries C that \mathcal{A} makes to $\mathcal{ABE}.\text{KeyGen}(\text{msk}, \cdot)$ satisfies $C(\mathbf{x}_i) = 0$ for all i (that is, sk_C does not decrypt the ciphertext corresponding to any of the \mathbf{x}_i). An attribute-based encryption scheme is selectively secure if for all PPT adversaries \mathcal{A} , the advantage $\text{Adv}_{\mathcal{A}}^{\text{ABE}}$ is a negligible function in λ .

We will use a special type of attribute-based encryption scheme with succinct keys, namely one where $|\text{sk}_C|$ does not grow with the size of the circuit C , but rather only its depth.

Theorem 3.4 ([BGG⁺14]). *Let λ be the security parameter, and $d \in \mathbb{N}$. Let $n = n(\lambda, d)$, $q = q(\lambda, d) = n^{O(d)}$, and let χ be a $\text{poly}(n)$ -bounded error distribution. Then, there is a selectively secure ABE scheme for the class of depth- d -bounded circuits, based on the hardness of $\text{DLWE}_{n,q,\chi}$. Furthermore, the secret key sk_C for a circuit C has size $\text{poly}(\lambda, n, d)$.*

4 Embedding Circuits into Matrices

In this section, we present the core techniques that we use in our construction. In essence, we use a method, developed in a recent work by Boneh et al. [BGG⁺14] to “embed” bits x_1, \dots, x_k into matrices $\mathbf{A}_1, \dots, \mathbf{A}_k$ and compute a circuit F on these matrices. This is done through a pair of algorithms (`ComputeA`, `ComputeC`) satisfying the following properties:

1. The deterministic algorithm `ComputeA` takes as input a circuit $F : \{0, 1\}^k \rightarrow \{0, 1\}$ and k matrices $\mathbf{A}_1, \dots, \mathbf{A}_k$, and outputs a matrix \mathbf{A}_F ; and
2. The deterministic algorithm `ComputeC` takes as input a bit string $\mathbf{x} = (x_1, \dots, x_k) \in \{0, 1\}^k$, and k LWE samples $\mathbf{s}^T(\mathbf{A}_i + x_i \mathbf{G}) + \mathbf{e}_i$, and outputs an LWE sample $\mathbf{s}^T(\mathbf{A}_F + F(\mathbf{x}) \cdot \mathbf{G}) + \mathbf{e}_F$ associated to the output matrix \mathbf{A}_F and the output bit $F(\mathbf{x})$.

These algorithms are closely modeled on the work of Boneh et al. [BGG⁺14]. We now describe how these algorithms work, and what their properties are.

The Algorithm `ComputeA`. Given a circuit F , input matrices $\mathbf{A}_1, \dots, \mathbf{A}_k$ (corresponding to the k input wires) and an auxiliary matrix \mathbf{A}_0 , the `ComputeA` procedure works inductively, going through the gates of the circuit F from the input to the output. Assume without loss of generality that the circuit F is composed of NOT and AND gates. For every AND gate $g = (u, v; w)$, assume inductively that we have computed matrices \mathbf{A}_u and \mathbf{A}_v for the input wires u and v . Define

$$\mathbf{A}_w = -\mathbf{A}_u \cdot \mathbf{G}^{-1}(\mathbf{A}_v)$$

For every NOT gate $g = (u; w)$, define

$$\mathbf{A}_w = \mathbf{A}_0 - \mathbf{A}_u$$

The Algorithm `ComputeC`. Given a circuit F , an input $x \in \{0, 1\}^k$ and LWE samples $(\mathbf{A}_i, \mathbf{y}_i)$, the `ComputeC` algorithm works as follows. For each AND gate $g = (u, v; w)$, assume that we have computed LWE samples $(\mathbf{A}_u, \mathbf{y}_u)$ and $(\mathbf{A}_v, \mathbf{y}_v)$ for the input wires u and v . Define

$$\mathbf{y}_w = x_u \cdot \mathbf{y}_v - \mathbf{y}_u \cdot \mathbf{G}^{-1}(\mathbf{A}_v)$$

where x_u and x_v are the bits on wires u and v when evaluating the circuit F on input x . For every NOT gate $g = (u; w)$, define

$$\mathbf{y}_w = \mathbf{y}_0 - \mathbf{y}_u$$

We will need the following lemma about the behavior of `ComputeA` and `ComputeC`. (We remind the reader that we use $\|\cdot\|$ to denote the ℓ_∞ norm).

Lemma 4.1. Let F be a depth- d Boolean circuit on k input bits, and let $x \in \{0,1\}^k$ be an input. Let $\mathbf{A}_0, \mathbf{A}_1, \dots, \mathbf{A}_k \in \mathbb{Z}_q^{n \times m}$ and $\mathbf{y}_0, \dots, \mathbf{y}_k \in \mathbb{Z}_q^m$ be such that

$$\|\mathbf{y}_i - \mathbf{s}^T(\mathbf{A}_i + x_i \mathbf{G})\| \leq B \quad \text{for } i = 0, 1, \dots, k.$$

for some $\mathbf{s} \in \mathbb{Z}_q^n$ and $B = B(\lambda)$. Let $\mathbf{A}_F \leftarrow \text{ComputeA}(F, \mathbf{A}_0, \dots, \mathbf{A}_k)$ and $\mathbf{y}_F \leftarrow \text{ComputeC}(F, x, \mathbf{A}_0, \dots, \mathbf{A}_k, \mathbf{y}_0, \dots)$. Then, $\|\mathbf{y}_F - \mathbf{s}^T(\mathbf{A}_F + F(\mathbf{x}) \cdot \mathbf{G})\| \leq m^{O(d)} \cdot B$.

Furthermore, \mathbf{y}_F is a “low-norm” linear function of $\mathbf{y}_0, \dots, \mathbf{y}_k$. That is, there are matrices $\mathbf{Z}_0, \dots, \mathbf{Z}_k$ (which depend on the function F , the input \mathbf{x} , and the input matrices $\mathbf{A}_0, \dots, \mathbf{A}_k$) such that $\mathbf{y}_F = \sum_{i=0}^k \mathbf{y}_i \mathbf{Z}_i$ and $\|\mathbf{Z}_i\| \leq m^{O(d)} \cdot B$.

Proof. We show this by induction on the levels of the circuit F , starting from the input. Consider two cases.

AND gate. Consider an AND gate $g = (u, v; w)$ where the input wires are at level L , and assume that $\mathbf{y}_u = \mathbf{s}^T(\mathbf{A}_u + x_u \mathbf{G}) + \mathbf{e}_u$ and $\mathbf{y}_v = \mathbf{s}^T(\mathbf{A}_v + x_v \mathbf{G}) + \mathbf{e}_v$, with $\|\mathbf{e}_u\|, \|\mathbf{e}_v\| \leq (m+1)^L \cdot B$. Now,

$$\begin{aligned} \mathbf{y}_w &= x_u \cdot \mathbf{y}_v - \mathbf{y}_u \cdot \mathbf{G}^{-1}(\mathbf{A}_v) \\ &= x_u \cdot (\mathbf{s}^T(\mathbf{A}_v + x_v \mathbf{G}) + \mathbf{e}_v) - \left(\mathbf{s}^T(\mathbf{A}_u + x_u \mathbf{G}) + \mathbf{e}_u \right) \cdot \mathbf{G}^{-1}(\mathbf{A}_v) \\ &= \mathbf{s}^T \left(x_u \mathbf{A}_v + x_u x_v \mathbf{G} - \mathbf{A}_u \mathbf{G}^{-1}(\mathbf{A}_v) - x_u \mathbf{A}_v \right) + \left(-\mathbf{e}_u \mathbf{G}^{-1}(\mathbf{A}_v) + x_u \mathbf{e}_v \right) \\ &= \mathbf{s}^T(\mathbf{A}_w + x_w \mathbf{G}) + \mathbf{e}_w \end{aligned}$$

where $\mathbf{A}_w = -\mathbf{A}_u \cdot \mathbf{G}^{-1}(\mathbf{A}_v)$, $x_w = x_u x_v$, and

$$\|\mathbf{e}_w\| \leq m \cdot \|\mathbf{e}_u\| + \|\mathbf{e}_v\| \leq (m+1) \cdot (m+1)^L \cdot B \leq (m+1)^{L+1} \cdot B$$

NOT gate. In a similar vein, for a NOT gate $g = (u; w)$, assume that $\mathbf{y}_u = \mathbf{s}^T(\mathbf{A}_u + x_u \mathbf{G}) + \mathbf{e}_u$, with $\|\mathbf{e}_u\| \leq (m+1)^L \cdot B$. Then,

$$\begin{aligned} \mathbf{y}_w &= \mathbf{y}_0 - \mathbf{y}_u = \mathbf{s}^T(\mathbf{A}_0 + \mathbf{G} - \mathbf{A}_u - x_u \mathbf{G}) + (\mathbf{e}_0 - \mathbf{e}_u) \\ &= \mathbf{s}^T(\mathbf{A}_w + (1 - x_u) \mathbf{G}) + \mathbf{e}_w \end{aligned}$$

where $\mathbf{A}_w = \mathbf{A}_0 - \mathbf{A}_u$, $x_w = 1 - x_u$, and

$$\|\mathbf{e}_w\| \leq \|\mathbf{e}_0\| + \|\mathbf{e}_u\| \leq B + (m+1)^L \cdot B \leq (m+1)^{L+1} \cdot B$$

Thus, $\mathbf{y}_F = \mathbf{s}^T \mathbf{A}_F + \mathbf{e}_F$ where $\|\mathbf{e}_F\| \leq m^{O(d)} \cdot B$. Furthermore, both transformations are linear functions on \mathbf{y}_u and \mathbf{y}_v , as required. \square

5 Constrained PRF

5.1 Construction

A family of functions $\mathcal{F} \subseteq (\{0,1\}^* \rightarrow \{0,1\})$ is z -uniform if each function $F \in \mathcal{F}$ can be described by a string in $\{0,1\}^z$ (we associate F with its description), and there exists a uniform circuit

family $\{\mathcal{U}_k\}_{k \in \mathbb{N}}$ such that $\mathcal{U}_k : \{0, 1\}^z \times \{0, 1\}^k \rightarrow \{0, 1\}$ such that for all $x \in \{0, 1\}^k$ it holds that $\mathcal{U}_k(F, x) = F(x)$. We assume for the sake of simplicity that the depth of \mathcal{U}_k grows monotonically with k and for all d we let k_d to be the maximal input size for which \mathcal{U}_k has depth at most d . We define \mathcal{F}_d to be such that $F \in \mathcal{F}$ is undefined for inputs of length $k > k_d$. We call such a family d -depth-bounded.

Our constrained PRF for a z -uniform d -depth-bounded family \mathcal{F} works as follows.

- **KeyGen**($1^\lambda, 1^z, 1^d$): The key generation algorithm takes as input the maximum size z and depth d of the constraining circuits. Let \mathcal{H} be a family of admissible hash functions (see Section 3.4) and let $\ell = \ell(\lambda)$ be the output length of hash functions in the family.

Let $n = n(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ be parameters chosen as described in Section 5.2 below, let $m = n \lceil \log q \rceil$.

Generate $z+2\ell+3$ matrices as follows: let \mathbf{A}_0 and \mathbf{A}_1 be the “input matrices”, let $\mathbf{B}_1, \mathbf{B}_2, \dots, \mathbf{B}_z$ be the “function matrices”, let $\mathbf{C}_1, \dots, \mathbf{C}_{2\ell}$ be the “partitioning matrices”, and let \mathbf{D} be an “auxiliary matrix”. All of these matrices are uniform in $\mathbb{Z}_q^{n \times m}$ (note that the “gadget matrix” \mathbf{G} has the same dimensions). In addition sample an admissible hash function $H \xleftarrow{\$} \mathcal{H}_\lambda$.

The public parameters consist of

$$\mathcal{PP} = (H, \mathbf{A}_0, \mathbf{A}_1, \mathbf{B}_1, \dots, \mathbf{B}_z, \mathbf{C}_1, \dots, \mathbf{C}_{2\ell}, \mathbf{D})$$

The seed of the PRF is a uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$.

- **Eval**($\mathbf{s}, \mathcal{PP}, \mathbf{x}$) takes as input the PRF seed \mathbf{s} , the public parameters \mathcal{PP} , and an input $x \in \{0, 1\}^k$ such that $k \leq k_d$ (i.e. \mathcal{U}_k is of depth $\leq d$), and works as follows.

Recall that $\mathcal{U}_k : \{0, 1\}^z \times \{0, 1\}^k \rightarrow \{0, 1\}$ is the universal circuit that takes a description of a function F and an input x and outputs $\mathcal{U}_k(F, x) = F(x)$. Let $\Pi : \{0, 1\}^{2\ell} \times \{0, 1\}^\ell \rightarrow \{0, 1\}$ denote the circuit that computes $\Pi(L, w) = \Pi_L(w)$ from Definition 3.5. Note that Π can be implemented by a binary circuit of depth $\log(\ell) + O(1)$.

Let (x_1, \dots, x_k) denote the bits of x . Let $w = H(x)$, and let w_1, \dots, w_ℓ be its bits. Compute

$$\mathbf{B}_\mathcal{U} \leftarrow \text{ComputeA}(\mathcal{U}_k, \mathbf{B}_1, \dots, \mathbf{B}_z, \mathbf{A}_{x_1}, \mathbf{A}_{x_2}, \dots, \mathbf{A}_{x_k}) \quad (1)$$

$$\mathbf{C}_\Pi \leftarrow \text{ComputeA}(\Pi, \mathbf{C}_1, \dots, \mathbf{C}_{2\ell}, \mathbf{A}_{w_1}, \mathbf{A}_{w_2}, \dots, \mathbf{A}_{w_\ell}) \quad (2)$$

and output

$$\text{PRF}_\mathbf{s}(\mathbf{x}) = \lfloor \mathbf{s}^T \mathbf{B}_\mathcal{U} \cdot \mathbf{G}^{-1}(\mathbf{C}_\Pi) \cdot \mathbf{G}^{-1}(\mathbf{D}) \rfloor_p$$

- **Constrain**($\mathbf{s}, \mathcal{PP}, F$) takes as input the PRF key \mathbf{s} and a circuit F (of size at most z) and does the following. Compute

$$\mathbf{a}_b = \mathbf{s}^T (\mathbf{A}_b + b \cdot \mathbf{G}) + \mathbf{e}_{1,b}^T \in \mathbb{Z}_q^m \text{ for } b \in \{0, 1\}$$

$$\mathbf{b}_i = \mathbf{s}^T (\mathbf{B}_i + f_i \cdot \mathbf{G}) + \mathbf{e}_{2,i}^T \in \mathbb{Z}_q^m \text{ for all } i \in [z]$$

where the vectors \mathbf{e} are drawn from an error distribution χ to be specified later (in Section 5.2).

The constrained seed K_F is the tuple $(\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}_1, \dots, \mathbf{b}_z) \in (\mathbb{Z}_q^m)^{z+2}$.

- $\text{ConstrainEval}(K_F, \mathcal{PP}, \mathbf{x})$ takes as input the constrained key K_F and an input \mathbf{x} . It computes

$$\mathbf{b}_{\mathcal{U}, \mathbf{x}} \leftarrow \text{ComputeC}\left(\mathcal{U}, (\mathbf{b}_1, \dots, \mathbf{b}_z, \mathbf{a}_{x_1}, \dots, \mathbf{a}_{x_k}), (f_1, \dots, f_z, x_1, \dots, x_k)\right)$$

and outputs $\lfloor \mathbf{b}_{\mathcal{U}, \mathbf{x}} \cdot \mathbf{G}^{-1}(\mathbf{C}_\Pi) \cdot \mathbf{G}^{-1}(\mathbf{D}) \rfloor_p$, where \mathbf{C}_Π is defined as above.

5.2 Setting the Parameters

Let us start by providing a typical parameter setting, and then explain how parameters can be modified and the effect on security.

Consider setting $n(\lambda, d) = (\lambda \cdot d)^c$, for a constant c that will be discussed shortly. We will set χ to be a discrete Gaussian distribution $D_{\mathbb{Z}, \alpha q}$ s.t. $\alpha q = \Theta(\sqrt{n})$. We define $n' = \lambda$ and let $p_1, \dots, p_{n'} = m^{O(d+\log \ell)}$ be all primes, and $p = \text{poly}(\lambda)$ (in fact, there is a lot of freedom in the choice of p , and it can be as large as $m^{O(d+\log \ell)}$ under the same asymptotic hardness). Finally, let $q = p \cdot (\alpha q) \cdot \prod_{i \in [n']} p_i = m^{n' \cdot O(d+\log \ell)} = 2^{\tilde{O}(\lambda \cdot d)} = 2^{\tilde{O}(n^{1/c})}$ (recall that $\ell = \text{poly}(\lambda)$).

This parameter setting translates into a PRF with $m = n \lceil \log q \rceil \cdot \Theta(\log \lambda)$ output bits per input, whose security is based (as we show in the next section) on the hardness of approximating lattice problems to within a factor of $2^{\tilde{O}(n^{1/c})}$.

Taking larger values of c will increase the hardness of the underlying lattice problem, but at the cost of considerably increasing the element sizes.

5.3 Security

Throughout this section, we let \mathcal{F} be a family of z -uniform functions and let d be a depth bound (both can depend on λ). We let $n = n(\lambda, d)$, $m = m(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ and the noise distributions $\chi = \chi(\lambda, d)$ be as defined in Section 5.2. We let \mathcal{H} be the family of admissible hash functions as described in Section 3.4, with range $\{0, 1\}^\ell$.

Theorem 5.1. *Let \mathcal{F} be a family of z -uniform functions and let d be a depth bound (both can depend on λ). Let $n = n(\lambda, d)$, $m = m(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ and the noise distributions $\chi = \chi(\lambda, d)$ be as defined in Section 5.2. Further let $m' = m \cdot (z + 2\ell + 3)$, and $\gamma = \omega(\sqrt{n \log \lambda}) \cdot p \cdot m^{O(d+\log \ell)}$. Assuming the hardness of $\text{DLWE}_{n, q, \chi}$, $\text{1D-SIS-R}_{q, p, \gamma, m'}$ and the admissible hash function family \mathcal{H} , the scheme $\text{CPRF} = (\text{KeyGen}, \text{Eval}, \text{Constrain}, \text{ConstrainEval})$ is a single-key secure selective-function secure constrained PRF for \mathcal{F} .*

We note that the hardness of all three assumptions translates to the worst case hardness of approximating lattice problems such as GapSVP and SIVP to within sub-exponential factors.

Proof. Let \mathcal{A} be a PPT selective-constraint adaptive-input adversary against $\text{CPRF}_{z, d}$. Let $t = \text{poly}(\lambda)$ be the (polynomial) number of input queries made by \mathcal{A} (w.l.o.g). Let ϵ be the advantage of \mathcal{A} in the constrained PRF game. We let $B = \alpha q \cdot \omega(\sqrt{\log \lambda})$. It holds that with all but negligible probabilities, all samples that we take from χ will have absolute value at most B . For the duration of the proof we assume that this is indeed the case.

The proof will proceed by a sequence of hybrids (or experiments) where the challenger samples a bit $b \in \{0, 1\}$ and interacts with \mathcal{A} . We let $\text{Adv}_H(\mathcal{A})$ denote the probability that \mathcal{A} outputs b in hybrid H .

Hybrid H_0 . This hybrid is the legitimate constrained PRF security game. The challenger generates $(s, \mathcal{PP}) \leftarrow \text{KeyGen}(1^\lambda, 1^z, 1^d)$. It gets $F \in \{0, 1\}^z$ from \mathcal{A} and produces a constrained key $K_F \leftarrow \text{Constrain}(s, \mathcal{PP}, F)$. It then sends \mathcal{PP}, K_F to \mathcal{A} . At this point \mathcal{A} adaptively makes queries $x^{(i)} \in \{0, 1\}^*$, and the challenger computes $y^{(i)} \leftarrow \text{Eval}(s, \mathcal{PP}, x^{(i)})$ and returns it to \mathcal{A} . Finally, \mathcal{A} outputs $x^* \in \{0, 1\}^*$. If $b = 0$ then the challenger returns $y^* \leftarrow \text{Eval}(s, \mathcal{PP}, x^*)$, and if $b = 1$ it returns a random y^* . Therefore, we have

$$\text{Adv}_{H_0}(\mathcal{A}) \geq 1/2 + \epsilon.$$

Hybrid H_1 . This is the notorious “artificial abort” phase. Let $\Delta_t = \Delta_t(\lambda)$ be the noticeable function from Definition 3.5. This hybrid is identical to the previous one, except in the last step the challenger flips a coin and with probability $1 - \Delta_t/2$ aborts the experiment (hence giving the adversary no information on b).

The adversary’s advantage thus degrades appropriately:

$$\text{Adv}_{H_1}(\mathcal{A}) \geq (\Delta_t/2) \cdot (1/2 + \epsilon) + (1 - \Delta_t/2) \cdot (1/2) = 1/2 + \epsilon \cdot \Delta_t/2.$$

Hybrid H_2 . In this hybrid, we associate some meaning with the artificial abort. Intuitively, the abort will be associated with a failure of the admissible hash function to partition the queries correctly. We are guaranteed that correct partitioning happens with probability $\geq \Delta_t$ (except for sequences that are hard to generate), but we would like to make it *(almost) exactly* $\Delta_t/2$ so as to not correlate the adversary’s success probability with the string L (the loss of the 2 factor is due to probability estimation).

Specifically, in this hybrid, rather than flipping a coin at the end of the experiment, the challenger does the following. For all $\vec{x} = (x^{(1)}, \dots, x^{(t)}, x^*)$, we define the event $\text{GoodPartition}_{L, \vec{x}}$ to be the event in which $\Pi_L(H(x^{(1)})) = \dots = \Pi_L(H(x^{(t)})) = 0$ and $\Pi_L(H(x^*)) = 1$, and define $\delta_{\vec{x}} = \Pr_{L \leftarrow \mathcal{L}_t}[\text{GoodPartition}_{\vec{x}, L}]$. The challenger will first compute an estimate $\tilde{\delta}_{\vec{x}}$ of $\delta_{\vec{x}}$ by sampling multiple values of L from \mathcal{L}_t and using Chernoff (both additive and multiplicative). Using $\text{poly}(\lambda)$ -many samples we can compute $\tilde{\delta}_{\vec{x}}$ such that

$$\Pr \left[\left| \delta_{\vec{x}} - \tilde{\delta}_{\vec{x}} \right| > \Delta_t/4 \right] \leq 2^{-\lambda}.$$

and in addition if $\delta_{\vec{x}} \geq \Delta_t/2$ then

$$\Pr \left[\left| \frac{\delta_{\vec{x}}}{\tilde{\delta}_{\vec{x}}} - 1 \right| > \epsilon/2 \right] \leq 2^{-\lambda}.$$

The challenger will then perform as follows: (i) It first verifies that $\tilde{\delta}_{\vec{x}} \geq \frac{3}{4}\Delta_t$, and aborts if this is not the case. (ii) It then samples $L \xleftarrow{\$} \mathcal{L}_t$ and aborts if $\text{GoodPartition}_{\vec{x}, L}$ did not occur (note that by our definitions above, this happens with probability $1 - \delta_{\vec{x}}$ over the choice of L). (iii) Then it flips a coin with probability $\frac{\tilde{\delta}_{\vec{x}} - \Delta_t/2}{\tilde{\delta}_{\vec{x}}}$ and aborts if the outcome is 1. Otherwise it carries out the experiment towards completion.

To analyze the effect on the success probability, we first notice that the probability that $\tilde{\delta}_{\vec{x}} < \frac{3}{4}\Delta_t$ (abortion is step (i)) is negligible. This is since, except with $2^{-\lambda}$ probability, this indicates that $\delta_{\vec{x}} < \Delta_t$, which implies that $\vec{x} \in \text{bad}_H$. Definition 3.5 guarantees that this happens with probability at most $\nu(\lambda) = \text{negl}(\lambda)$.

If the above abort did not occur, we know that $\delta_{\tilde{x}} \geq \Delta_t/2$ (except with probability $2^{-\lambda}$), we first notice that the total probability of abort in steps (ii) + (iii)

$$1 - \delta_{\tilde{x}} + \delta_{\tilde{x}} \cdot \frac{\tilde{\delta}_{\tilde{x}} - \Delta_t/2}{\tilde{\delta}_{\tilde{x}}} = 1 - \frac{\delta_{\tilde{x}}}{\tilde{\delta}_{\tilde{x}}} \Delta_t/2 \in [(1 - \Delta_t/2) - \epsilon\Delta_t/4, (1 - \Delta_t/2) + \epsilon\Delta_t/4]$$

It therefore follows that if there was no abort in step (i), then the adversary's view in H_2 is within statistical distance $2^{-\lambda} + \epsilon\Delta_t/4$ from its view in H_1 .

Putting all steps together, we get that

$$\text{Adv}_{H_2}(\mathcal{A}) \geq 1/2 + \epsilon \cdot \Delta_t/2 - \nu(\lambda) - O(2^{-\lambda}) - \epsilon\Delta_t/4 = 1/2 + \epsilon \cdot \Delta_t/4 - \text{negl}(\lambda) .$$

Hybrid H_3 . In this hybrid, the challenger first samples $L \xleftarrow{\$} \mathcal{L}_t$, and then, for each $x^{(i)}$ in turn, it checks whether $\Pi_L(H(x^{(i)})) = 0$, and immediately aborts if not. Similarly, upon receiving x^* , it checks whether $\Pi_L(H(x^*)) = 1$ and immediately aborts if not. Otherwise it continues the same as H_2 .

It is rather straightforward to see that the \mathcal{A} 's advantage does not change. The cases in which we abort are exactly the same as the ones in the previous hybrid (since it is sufficient that a single $x^{(i)}$ does not give the required value in order to abort). Further, the sampling of L has been completely independent of all the other randomness in the experiment so it might as well happen in the beginning. We conclude that

$$\text{Adv}_{H_3}(\mathcal{A}) = \text{Adv}_{H_2}(\mathcal{A}) \geq 1/2 + \epsilon \cdot \Delta_t/4 - \text{negl}(\lambda) .$$

Hybrid H_4 . In this hybrid, the challenger changes the way the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are generated. Recall that our security game is constraint-selective, namely \mathcal{A} produces the constraint F before seeing the public parameters.

Therefore, here, the challenger waits until receiving F from \mathcal{A} and only generates the public parameters at that point (note that by then L has also been specified). To generate the public parameters, the matrix \mathbf{D} is produced identically to before. In addition, the challenger samples matrices $\{\hat{\mathbf{A}}_\beta\}_{\beta \in \{0,1\}}, \{\hat{\mathbf{B}}_i\}_{i \in [z]}, \{\hat{\mathbf{C}}_i\}_{i \in [2\ell]}$. It then sets

$$\mathbf{A}_\beta = \hat{\mathbf{A}}_\beta - \beta \mathbf{G}$$

$$\mathbf{B}_i = \hat{\mathbf{B}}_i - f_i \mathbf{G}$$

$$\mathbf{C}_i = \hat{\mathbf{C}}_i - L_i \mathbf{G}$$

The remainder of the experiment remains unchanged.

Since the distributions of the $\mathbf{A}, \mathbf{B}, \mathbf{C}$ matrices is identical to their original uniform distributions, it follows that

$$\text{Adv}_{H_4}(\mathcal{A}) = \text{Adv}_{H_3}(\mathcal{A}) .$$

Hybrid H_5 . In this hybrid, the adversary changes the way it computes the outputs $y^{(i)}$. Recall that $K_F = (\mathbf{a}_0, \mathbf{a}_1, \mathbf{b}_1, \dots, \mathbf{b}_z)$ is the constrained key given to \mathcal{A} . Let us denote

$$\mathbf{c}_i = \mathbf{s}^T (\mathbf{C}_i + L_i \mathbf{G}) + \mathbf{e}_{3,i}^T \quad \text{for all } i \in [z]$$

$$\mathbf{d} = \mathbf{s}^T \mathbf{D} + \mathbf{e}_4^T$$

where $\mathbf{e}_{3,i}$ are sampled coordinate-wise from χ , and \mathbf{e}_4 is sampled coordinate-wise from χ' .

In this hybrid, in order to answer input queries, the challenger first computes

$$\mathbf{b}_{\mathcal{U},x^{(i)}} \leftarrow \text{ComputeC}\left(\mathcal{U}, (\mathbf{b}_1, \dots, \mathbf{b}_z, \mathbf{a}_{x_1}, \dots, \mathbf{a}_{x_k}), (f_1, \dots, f_z, x_1^{(i)}, \dots, x_k^{(i)})\right)$$

and then, letting $w^{(i)} = H(x^{(i)})$

$$\mathbf{c}_{\Pi,w^{(i)}} \leftarrow \text{ComputeC}\left(\Pi, (\mathbf{c}_1, \dots, \mathbf{c}_{2\ell}, \mathbf{a}_{w_1}, \dots, \mathbf{a}_{w_\ell}), (L_1, \dots, L_{2\ell}, w_1^{(i)}, \dots, w_\ell^{(i)})\right)$$

We recall that by Lemma 4.1 it holds that

$$\begin{aligned}\mathbf{b}_{\mathcal{U},x^{(i)}}^T &= \mathbf{s}^T(\mathbf{B}_{\mathcal{U},x^{(i)}} + F(x^{(i)}) \cdot \mathbf{G}) + \mathbf{e}_{\mathcal{U}}^T \\ \mathbf{c}_{\Pi,w^{(i)}}^T &= \mathbf{s}^T(\mathbf{C}_{\Pi,w^{(i)}} + \Pi_L(w^{(i)}) \cdot \mathbf{G}) + \mathbf{e}_{\Pi}^T,\end{aligned}$$

for some $\mathbf{e}_{\mathcal{U}}, \mathbf{e}_{\Pi}$ for which $\|\mathbf{e}_{\mathcal{U}}\| \leq B \cdot m^{O(d)}$, $\|\mathbf{e}_{\Pi}\| \leq B \cdot m^{O(\log \ell)}$.

We recall that by definition

$$\begin{aligned}\text{PRF}_{\mathbf{s}}(x^{(i)}) &= \left[\mathbf{s}^T \mathbf{B}_{\mathcal{U},x^{(i)}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) \right]_p \\ &= \left[\mathbf{s}^T (\mathbf{B}_{\mathcal{U},x^{(i)}} + F(x^{(i)}) \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) \right. \\ &\quad \left. - F(x^{(i)}) \mathbf{s}^T \mathbf{C}_{\Pi,w^{(i)}} \mathbf{G}^{-1}(\mathbf{D}) \right]_p \\ &= \left[\mathbf{s}^T (\mathbf{B}_{\mathcal{U},x^{(i)}} + F(x^{(i)}) \mathbf{G}) \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) \right. \\ &\quad \left. - F(x^{(i)}) \mathbf{s}^T (\mathbf{C}_{\Pi,w^{(i)}} + \Pi_L(w^{(i)}) \mathbf{G}) \mathbf{G}^{-1}(\mathbf{D}) \right. \\ &\quad \left. + F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{s}^T \mathbf{D} \right]_p \\ &= \left[\mathbf{b}_{\mathcal{U},x^{(i)}}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)}) \mathbf{c}_{\Pi,w^{(i)}}^T \mathbf{G}^{-1}(\mathbf{D}) \right. \\ &\quad \left. + F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{d}^T + \mathbf{e}'^T \right]_p,\end{aligned}\tag{3}$$

where

$$\mathbf{e}'^T = -\mathbf{e}_{\mathcal{U}}^T \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) + F(x^{(i)}) \mathbf{e}_{\Pi}^T \mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{e}_4^T\tag{4}$$

which implies that $\|\mathbf{e}'\| \leq E$ for some $E = (m^{O(d)} + m^{O(\log \ell)}) \cdot B$.

To analyze the distinguishing probability between these hybrids, for any input x (and $w = H(x)$) we define the event Borderline_x as the event where there exists $j \in [m]$ such that:

$$\begin{aligned}(\mathbf{b}_{\mathcal{U},x}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w}) \cdot \mathbf{G}^{-1}(\mathbf{D}) - F(x) \cdot \mathbf{c}_{\Pi,w}^T \cdot \mathbf{G}^{-1}(\mathbf{D}) \\ + F(x) \cdot \Pi_L(w) \cdot \mathbf{d}^T) \cdot \mathbf{u}_j \in [-E, E] + (q/p)\mathbb{Z},\end{aligned}$$

where we recall that \mathbf{u}_j is the j th indicator vector. Namely, this is the probability that one of the coordinates of the vector $\mathbf{b}_{\mathcal{U},x}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w}) \mathbf{G}^{-1}(\mathbf{D}) - F(x) \mathbf{c}_{\Pi,w}^T \mathbf{G}^{-1}(\mathbf{D}) + F(x) \Pi_L(w) \mathbf{d}^T$ is “dangerously close” to being rounded in the wrong direction.

By definition of rounding, if $\neg \text{Borderline}_{x^{(i)}}$, then

$$\begin{aligned} \text{PRF}_{\mathbf{s}}(x^{(i)}) &= \lfloor \mathbf{b}_{\mathcal{U},x^{(i)}}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)}) \mathbf{c}_{\Pi,w^{(i)}}^T \mathbf{G}^{-1}(\mathbf{D}) \\ &\quad + F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{d}^T \rfloor_p . \end{aligned}$$

The challenger in this hybrid, given a query $x^{(i)}$, will first check whether $\text{Borderline}_{x^{(i)}}$. If the event happens, the challenger aborts. Otherwise it returns $\text{PRF}_{\mathbf{s}}(x^{(i)})$ as defined above. Note that the challenger only needs to respond to queries $x^{(i)}$ for which $\Pi_L(w^{(i)}) = \Pi_L(H(x^{(i)})) = 0$, which do not depend on \mathbf{d} , a fact that will be important later on.

Finally, on the challenge query x^* , unless abort is needed, it holds that $F(x^*) = 1$ and $\Pi_L(w^*) = 1$ (where $w^* = H(x^*)$) and therefore, unless the event Borderline_{x^*} happens, it holds that

$$\text{PRF}_{\mathbf{s}}(x^*) = \left\lfloor \mathbf{b}_{\mathcal{U},x^*}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) - \mathbf{c}_{\Pi,w^*}^T \mathbf{G}^{-1}(\mathbf{D}) + \mathbf{d}^T \right\rfloor_p .$$

The challenger will therefore abort if Borderline_{x^*} and return the aforementioned value otherwise (that is if the bit b is 0; if $b = 1$ then of course a uniform value is returned).

It follows that if we define $\text{Borderline} = (\bigvee_i \text{Borderline}_{x^{(i)}}) \vee \text{Borderline}_{x^*}$, then

$$|\text{Adv}_{\text{H}_5}(\mathcal{A}) - \text{Adv}_{\text{H}_4}(\mathcal{A})| \leq \Pr_{\text{H}_5}[\text{Borderline}] .$$

We will bound $\Pr_{\text{H}_5}[\text{Borderline}]$ as a part of our analysis in the next hybrid.

As a final remark on this hybrid, we note that in order to execute this hybrid, the challenger does not need to access \mathbf{s} itself, but rather only the $\mathbf{a}_\beta, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}$ vectors. This will be useful in the next hybrid.

Hybrid H_6 . In this hybrid, all $\mathbf{a}_\beta, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}$ are sampled from the uniform distribution. Everything else remains the same. We note that by definition, in hybrid H_5 :

$$\begin{aligned} \mathbf{a}_\beta^T &= \mathbf{s}^T \hat{\mathbf{A}}_\beta + \mathbf{e}_{1,\beta}^T \\ \mathbf{b}_i^T &= \mathbf{s}^T \hat{\mathbf{B}}_i + \mathbf{e}_{2,i}^T \\ \mathbf{c}_i^T &= \mathbf{s}^T \hat{\mathbf{C}}_i + \mathbf{e}_{3,i}^T \\ \mathbf{d}^T &= \mathbf{s}^T \mathbf{D} + \mathbf{e}_4^T , \end{aligned}$$

where all $\hat{\mathbf{A}}_\beta, \hat{\mathbf{B}}_i, \hat{\mathbf{C}}_i, \mathbf{D}$ are uniformly distributed, and all $\mathbf{e}_{1,\beta}^T, \mathbf{e}_{2,i}^T, \mathbf{e}_{3,i}^T, \mathbf{e}_4^T$ are sampled coordinate-wise from χ . The $\text{DLWE}_{n,q,\chi}$ assumption therefore asserts that:

$$|\text{Adv}_{\text{H}_6}(\mathcal{A}) - \text{Adv}_{\text{H}_5}(\mathcal{A})| \leq \text{negl}(\lambda) .$$

Furthermore, since Borderline is an efficiently recognizable event, it also holds that

$$\left| \Pr_{\text{H}_6}[\text{Borderline}] - \Pr_{\text{H}_5}[\text{Borderline}] \right| = \text{negl}(\lambda) . \quad (5)$$

In H_6 , the probability of Borderline can be bounded under the 1D-SIS-R assumption.

Claim 5.1.1. *Under the 1D-SIS- $\text{R}_{q,p,\gamma,m'}$ assumption, it holds that $\Pr_{\text{H}_6}[\text{Borderline}] = \text{negl}(\lambda)$, where $m' = m \cdot (2 + z + 2\ell + 1)$, and $\gamma = p \cdot B \cdot m^{O(d+\log \ell)}$.*

Proof. Let $\mathbf{v} \in \mathbb{Z}_q^{(2+z+2\ell+1)m}$ be an input to 1D-SIS- $R_{q,p,\gamma,m'}$. Then define $\mathbf{a}_\beta, \mathbf{b}_i, \mathbf{c}_i, \mathbf{d}$ be so that their concatenation is \mathbf{v} .

The reduction executes H_6 as the challenger, using the vectors defined above. We claim that if **Borderline** occurs, then we solve 1D-SIS-R. This follows since if **Borderline** occurs then we found x, j such that

$$\begin{aligned} & (\mathbf{b}_{\mathcal{U},x}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w})\mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)})\mathbf{c}_{\Pi,w}^T\mathbf{G}^{-1}(\mathbf{D}) + F(x)\Pi_L(w)\mathbf{d}^T)\mathbf{u}_j \\ & \in [-E, E] + (q/p)\mathbb{Z} . \end{aligned}$$

However, by Lemma 4.1, it follows that

$$\begin{aligned} \mathbf{b}_{\mathcal{U},x}^T &= \sum_{\beta \in \{0,1\}} \mathbf{a}_\beta^T \mathbf{R}'_{1,\beta} + \sum_{i \in [z]} \mathbf{b}_i^T \mathbf{R}'_{2,i} \\ \mathbf{c}_{\Pi,x}^T &= \sum_{\beta \in \{0,1\}} \mathbf{a}_\beta^T \mathbf{R}''_{1,\beta} + \sum_{i \in [2\ell]} \mathbf{c}_i^T \mathbf{R}''_{3,i} \end{aligned}$$

where $\|\mathbf{R}'_{1,\beta}\|, \|\mathbf{R}'_{2,i}\| \leq m^{O(d)}$ and $\|\mathbf{R}''_{1,\beta}\|, \|\mathbf{R}''_{3,i}\| \leq m^{O(\log \ell)}$. It follows that there exists an (efficiently derivable) matrix \mathbf{R}_0 such that

$$\mathbf{b}_{\mathcal{U},x}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w})\mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)})\mathbf{c}_{\Pi,w}^T\mathbf{G}^{-1}(\mathbf{D}) + F(x)\Pi_L(w)\mathbf{d}^T = \mathbf{v}^T \mathbf{R}_0 ,$$

and $\|\mathbf{R}_0\| \leq m^{O(d+\log \ell)}$.

Finally,

$$\langle \mathbf{v}, \mathbf{R}_0 \cdot \mathbf{u}_j \rangle \in [-E, E] + (q/p)\mathbb{Z} ,$$

with $\|\mathbf{R}_0 \cdot \mathbf{u}_j\| \leq \|\mathbf{R}_0\| \leq m^{O(d+\log \ell)}$ and $E = B \cdot m^{O(d+\log \ell)} = m^{O(d+\log \ell)}$. Thus $\mathbf{R}_0 \cdot \mathbf{u}_j$ is a valid solution for 1D-SIS- $R_{q,p,\gamma,m'}$. The claim thus follows. \blacksquare

Putting together Claim 5.1.1 and Eq. (5), we get that

$$\Pr_{H_5}[\text{Borderline}] \leq \Pr_{H_6}[\text{Borderline}] + \text{negl}(\lambda) \leq \text{negl}(\lambda) .$$

and thus, finally

$$|\text{Adv}_{H_5}(\mathcal{A}) - \text{Adv}_{H_6}(\mathcal{A})| \leq \text{negl}(\lambda) .$$

Finally, we notice that the vector \mathbf{d} is only used when answering the challenge query in the case of $b = 0$. This means that in the adversary's view, the answer it gets when $b = 0$ is uniform and independent of its view so far, exactly the same as the case $b = 1$ where an actual random vector is returned. It follows that

$$\text{Adv}_{H_6}(\mathcal{A}) = 1/2 .$$

On the other hand

$$\text{Adv}_{H_6}(\mathcal{A}) \geq 1/2 + \epsilon \Delta_t/4 - \text{negl}(\lambda) ,$$

and thus

$$\epsilon \leq \frac{\text{negl}(\lambda)}{\Delta_t/4} = \text{negl}(\lambda) .$$

It follows that \mathcal{A} cannot achieve a noticeable advantage in the constrained PRF experiment under the DLWE $_{q,n,\chi}$ assumption. \square

5.4 Computational Functionality Preserving

We now prove the computational functionality preservation of our scheme, as per Definition 3.1. Throughout this section, we let \mathcal{F} be a family of z -uniform functions and let d be a depth bound (both can depend on λ). We let $n = n(\lambda, d)$, $m = m(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ and the noise distributions $\chi = \chi(\lambda, d)$ be as defined in Section 5.2. We let \mathcal{H} be the family of admissible hash functions as described in Section 3.4, with range $\{0, 1\}^\ell$.

Theorem 5.2. *Let \mathcal{F} be a family of z -uniform functions and let d be a depth bound (both can depend on λ). Let $n = n(\lambda, d)$, $m = m(\lambda, d)$, $q = q(\lambda, d)$, $p = p(\lambda, d)$ and the noise distributions $\chi = \chi(\lambda, d)$ be as defined in Section 5.2. Further let $m' = m \cdot (z + 2\ell + 3)$, and $\gamma = \omega(\sqrt{n \log \lambda}) \cdot p \cdot m^{O(d + \log \ell)}$.*

Assuming the hardness of $\text{DLWE}_{n,q,\chi}$ and $\text{1D-SIS-R}_{q,p,\gamma,m'}$, the scheme \mathcal{CPRF} is computationally functionality preserving.

We note that the hardness of both assumptions translates to the worst case hardness of approximating lattice problems such as GapSVP and SIVP to within sub-exponential factors.

outline. The theorem follows from an argument practically identical to that made in Hybrids H_5, H_6 of the proof of Theorem 5.1.

Recall that we showed that Borderline events only happen with negligible probability, and therefore with all but negligible probability, it holds that the PRF value at point $x^{(i)}$ is exactly equal to

$$\left[\mathbf{b}_{\mathcal{U},x^{(i)}}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) - F(x^{(i)}) \mathbf{c}_{\Pi,w^{(i)}}^T \mathbf{G}^{-1}(\mathbf{D}) + F(x^{(i)}) \Pi_L(w^{(i)}) \mathbf{d}^T \right]_p.$$

However, when $F(x^{(i)}) = 0$, this term simplifies to

$$\left[\mathbf{b}_{\mathcal{U},x^{(i)}}^T \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi,w^{(i)}}) \mathbf{G}^{-1}(\mathbf{D}) \right]_p$$

which is exactly $\text{ConstrainEval}(K_F, \mathcal{PP}, x^{(i)})$ by definition. Functionality is thus preserved with all but negligible probability. \square

5.5 Other Properties

We describe several other properties that our construction satisfies.

Unconditional Almost-Correctness. We have shown that our constrained PRF satisfies a computational correctness property, namely that it is hard to find an input \mathbf{x} such that $\text{PRF}_K(\mathbf{x}) \neq \text{ConstrainEval}(K_F, \mathcal{PP}, \mathbf{x})$. We are also able to show unconditionally that the constrained evaluation and the actual PRF evaluation do not differ by much, for any input \mathbf{x} . Indeed, by Equation 3 and 4, we have

$$\|\text{PRF}_K(\mathbf{x}) - \text{ConstrainEval}(K_F, \mathcal{PP}, \mathbf{x})\|_\infty \leq m^{O(d)} \cdot B$$

Key Homomorphism. Our PRF is also “almost key homomorphic” in the sense that $\text{PRF}_{\mathbf{s}}(\mathbf{x}) + \text{PRF}_{\mathbf{s}'}(\mathbf{x})$ is close to $\text{PRF}_{\mathbf{s}+\mathbf{s}'}(\mathbf{x})$ for any keys \mathbf{s} and \mathbf{s}' and any input \mathbf{x} . Recall that our PRF is

$$\text{PRF}_{\mathbf{s}}(\mathbf{x}) = \left[\mathbf{s}^T \mathbf{B}_{\mathcal{U}} \cdot \mathbf{G}^{-1}(\mathbf{C}_{\Pi}) \cdot \mathbf{G}^{-1}(\mathbf{D}) \right]_p$$

For any keys \mathbf{s}_i and input \mathbf{x} , denoting $\mathbf{s}_i^T \mathbf{B}_U \cdot \mathbf{G}^{-1}(\mathbf{C}_\Pi) \cdot \mathbf{G}^{-1}(\mathbf{D})$ as \mathbf{h}_i , we have

$$\|\text{PRF}_{\sum \mathbf{s}_i}(\mathbf{x}) - \sum \text{PRF}_{\mathbf{s}_i}(\mathbf{x})\|_\infty = \left\| \left[\sum_i \mathbf{h}_i \right]_p - \sum_i [\mathbf{h}_i]_p \right\|_\infty \leq k + 1$$

Constrained-Key Homomorphism. Our constrained keys are “almost homomorphic” as well, in the same sense as above. That is, if K_F and K'_F are constrained versions of PRF keys K and K' for the same function F , the summation $K_F + K'_F$ is a constrained version of $K + K'$ for the function F . For any input \mathbf{x} , we then have that $\text{ConstrainEval}(K_F + K'_F, \mathcal{PP}, \mathbf{x})$ is close to $\text{PRF}_{K+K'}(\mathbf{x})$.

We remark that techniques similar to what we used in showing computational correctness can be used to strengthen the almost key-homomorphism property into computational key-homomorphism where it is computationally hard to find an input for which key homomorphism does not hold.

6 Succinct Constrained Keys

In this section we show how to reduce the size of the constrained key so that asymptotically it depends only on the security parameter and independent of the function class. The construction builds upon the scheme \mathcal{CPRF} from Section 5 but reduces the key size by utilizing an attribute based encryption scheme (ABE). In particular, the constrained keys in our new system have size $\text{poly}(\lambda)$, independent of the parameters of the constraining circuit (namely, its size or depth).

Our *succinct* constrained PRF \mathcal{SCPRF} for a z -uniform d -depth-bounded family \mathcal{F} works as follows.

- **KeyGen**($1^\lambda, 1^z, 1^d$): The key generation algorithm takes as input the maximum size z and depth d of the constraining circuits. Let $t = O(\log z)$ to be specified later.

It starts by calling $\mathcal{CPRF}.\text{KeyGen}(1^\lambda, 1^z, 1^d)$ to obtain the seed \mathbf{s} , and public parameters $\mathcal{PP} = (H, \mathbf{A}_0, \mathbf{A}_1, \{\mathbf{B}_i\}_{i \in [z]})$.

It then generates: $\mathbf{a}_\beta = \mathbf{s}^T(\mathbf{A}_\beta + \beta \mathbf{G}) + \mathbf{e}_{1,\beta}^T$ and $\mathbf{b}_{i,\beta} = \mathbf{s}^T(\mathbf{B}_i + \beta \mathbf{G}) + \mathbf{e}_{2,i,\beta}^T$. Note that any possible constrained key of \mathcal{CPRF} consists of \mathbf{a}_0 and \mathbf{a}_1 , together with a subset of $\{\mathbf{b}_{i,\beta}\}_{i \in [z], \beta \in \{0,1\}}$.

Next it generates parameters for the ABE scheme $(\mathcal{ABE}.\text{msk}, \mathcal{ABE}.\text{pp}) \leftarrow \mathcal{ABE}.\text{Setup}(1^\lambda, 1^t)$, and generates $\text{ct}_{i,\beta} \leftarrow \mathcal{ABE}.\text{Enc}(\mathcal{ABE}.\text{pp}, (i, \beta), \mathbf{b}_{i,\beta})$, encryptions with (i, β) as the “attributes” and $\mathbf{b}_{i,\beta}$ as the “message”.

The public parameters consist of

$$\mathcal{SCPRF}.\mathcal{PP} = (\mathcal{CPRF}.\mathcal{PP}, \mathcal{ABE}.\mathcal{PP}, \mathbf{a}_0, \mathbf{a}_1, \{\text{ct}_{i,\beta}\}_{i,\beta})$$

The seed for \mathcal{SCPRF} contains a seed for \mathcal{CPRF} , namely a uniformly random vector $\mathbf{s} \in \mathbb{Z}_q^n$, and in addition the ABE master secret key $\mathcal{ABE}.\text{msk}$. We note that in fact \mathbf{s} can be retrieved from the public parameters using $\mathcal{ABE}.\text{msk}$ and therefore it is not necessary to give it explicitly. However, it is more natural to think of \mathbf{s} as a part of the seed. In particular \mathbf{s} will be used to evaluate \mathcal{SCPRF} (see Eval below) and $\mathcal{ABE}.\text{msk}$ will be used to produce constrained keys (see Constrain below).

- $\text{Eval}(\mathbf{s}, \mathcal{PP}, \mathbf{x})$ takes as input the PRF seed \mathbf{s} , the public parameters \mathcal{PP} which contains $\mathcal{CPRF}.\text{pp}$, and an input $x \in \{0, 1\}^k$ such that $k \leq k_d$ (i.e. \mathcal{U}_k is of depth $\leq d$), and outputs the result of the CPRF evaluation, namely $\mathcal{CPRF}.\text{Eval}(\mathbf{s}, \mathcal{CPRF}.\text{pp}, \mathbf{x})$.
- $\text{Constrain}(\mathcal{ABE}.\text{msk}, F)$ takes as input the ABE master secret key $\mathcal{ABE}.\text{msk}$ and a circuit F (represented as a string in $\{0, 1\}^z$) and does the following. Consider the function:

$$\phi_F(i, \beta) = \begin{cases} 1, & \text{if } F_i = \beta \\ 0, & \text{otherwise} \end{cases}$$

Note that ϕ_F can be computed by a depth $O(\log z)$ circuit (whose depth is independent of the depth of F itself), the parameter t from above is set to be equal to this depth. We recall Section 3.5

The constrained key for F is the ABE token for ϕ_F , namely

$$K_F = \mathcal{ABE}.\text{KeyGen}(\mathcal{ABE}.\text{msk}, \phi_F)$$

- $\text{ConstrainEval}(K_F, \mathcal{PP}, \mathbf{x})$ takes as input the constrained key K_F , the public parameters \mathcal{PP} and an input \mathbf{x} .

Recalling that $\mathcal{PP} = (\mathcal{CPRF}.\text{pp}, \mathcal{ABE}.\text{pp}, \mathbf{a}_0, \mathbf{a}_1, \{\text{ct}_{i,\beta}\})$, and that K_F is the ABE decryption key for the function ϕ_F , it first decrypts to obtain $\mathbf{b}_i = \mathcal{ABE}.\text{Dec}(K_F, \text{ct}_{i,F_i})$, and then applies the constrained evaluation algorithm $\mathcal{CPRF}.\text{ConstrainEval}((\mathbf{a}_0, \mathbf{a}_1, \{\mathbf{b}_i\}), \mathcal{CPRF}.\mathcal{PP}, \mathbf{x})$.

The correctness follows in a straightforward manner from the correctness of \mathcal{ABE} and \mathcal{CPRF} . The constrained key size of \mathcal{SCPRF} is derived from that of \mathcal{ABE} and is $\text{poly}(\lambda, t) = \text{poly}(\lambda, \log z)$. It follows that there exists a $\text{poly}(\lambda)$ asymptotic upper bound on the key sizes that applies for all polynomial values of z . Security is proven in the following theorem.

Theorem 6.1. *If \mathcal{CPRF} is a single-key secure constrained pseudorandom function for function class \mathcal{F} (Definition 3.1), which is built according to the template in Section 5, and if \mathcal{ABE} is a selectively secure ABE scheme (Definition 3.6), then the scheme \mathcal{SCPRF} described above is a secure single-key CPRF for \mathcal{F} .*

Proof. Let \mathcal{A} be a CPRF adversary against \mathcal{SCPRF} . The proof will proceed by a sequence of hybrids where in each hybrid the challenger will sample a random bit b and the adversary's success in inferring b will be \mathcal{A} 's advantage in the hybrid.

Hybrid H_0 . This hybrid is the constrained PRF security game for \mathcal{SCPRF} . The challenger generates $(\mathcal{SK}, \mathcal{PP}) \leftarrow \text{KeyGen}(1^\lambda, 1^z, 1^d)$. It gets $F \in \{0, 1\}^z$ from \mathcal{A} and produces a constrained key $K_F \leftarrow \text{Constrain}(\mathcal{SK}, \mathcal{PP}, F)$. It then sends \mathcal{PP}, K_F to \mathcal{A} . At this point \mathcal{A} adaptively makes queries $x^{(i)} \in \{0, 1\}^*$, and the challenger computes $y^{(i)} \leftarrow \text{Eval}(\mathbf{s}, \mathcal{PP}, x^{(i)})$ and returns it to \mathcal{A} . Finally, \mathcal{A} outputs $x^* \in \{0, 1\}^*$. If $b = 0$ then the challenger returns $y^* \leftarrow \text{Eval}(\mathbf{s}, \mathcal{PP}, x^*)$, and if $b = 1$ it returns a random y^* .

$$\text{Adv}_{H_0}(\mathcal{A}) \geq 1/2 + \epsilon.$$

Hybrid H_1 . In this hybrid, the challenger does the following. It first receives the function F and then generates (SK, PP) with one change compared to the previous hybrid. The ciphertexts $ct_{i,1-F_i}$ will now be generated as $ABE.Enc(ABE.pp, (i, \beta), 0)$.

Claim 6.1.1. *Under the selective security of ABE , it holds that*

$$|\text{Adv}_{H_0}(\mathcal{A}) - \text{Adv}_{H_1}(\mathcal{A})| \leq \text{negl}(\lambda) ,$$

Proof. Let \mathcal{B} be the following adversary against multi-message selective security of ABE (see Definition 3.6), which will work by simulating the interaction of \mathcal{A} in the CPRF security game against $SCPRF$. First of all \mathcal{B} generates a key pair $(s, CPRF.PP) \leftarrow CPRF.KeyGen(1^\lambda, 1^z, 1^d)$, and produces the vectors $\mathbf{a}_\beta, \mathbf{b}_{i,\beta}$ as in the key generation process of $SCPRF$.

Then \mathcal{B} runs \mathcal{A} to obtain the function description $F \in \{0, 1\}^z$. It sends to the ABE challenger the attribute sequence $\{(i, 1 - F_i)\}_{i \in [z]}$. Then, for each i , it will send to the ABE challenger the message pair $m_{0,i} = \mathbf{b}_{i,1-F_i}, m_{1,i} = 0$. It receives $ABE.PP$ and ciphertexts $ct_{i,1-F_i}$ which encrypt either $m_{0,i}$ or $m_{1,i}$. Using $ABE.PP$ it generates ct_{i,F_i} by itself as in $SCPRF.KeyGen$. Further, \mathcal{B} generates $SCPRF.PP$ using $CPRF.PP, ABE.PP$ and $ct_{i,\beta}$, and forwards this value to \mathcal{A} . Note that this is distributed identically to an $SCPRF.PP$ in H_0 if $b = 0$ and identically to $SCPRF.PP$ in H_1 if $b = 1$.

Next, \mathcal{B} queries the ABE challenger on the function token ϕ_F , noting that $\phi_F(i, 1 - F_i) = 0$ for all i . The challenger responds with the appropriate token, which will be forwarded to \mathcal{A} as K_F . Note that this value is correctly distributed.

The adversary \mathcal{B} continues to simulate \mathcal{A} , answering its oracle queries using the seed s . Finally, when \mathcal{A} halts and outputs some b' , \mathcal{B} halts as well and outputs b' as its own output.

By definition, the advantage of \mathcal{B} against ABE is exactly $\text{Adv}_{H_0}(\mathcal{A}) - \text{Adv}_{H_1}(\mathcal{A})$, and the claim follows from the selective security of ABE . \blacksquare

Next, we notice that the adversary's advantage in this hybrid cannot be noticeable without breaking the security of $CPRF$.

Claim 6.1.2. *If $CPRF$ is a secure single-key constrained PRF then $|\text{Adv}_{H_1}(\mathcal{A}) - 1/2| = \text{negl}(\lambda)$.*

Proof. We present an adversary \mathcal{B} against $CPRF$ whose advantage is $|\text{Adv}_{H_1}(\mathcal{A}) - 1/2|$ as follows. It will first get F from \mathcal{A} and forward it to the $CPRF$ challenger. Then, upon receiving $CPRF.PP, \mathbf{a}_\beta, \mathbf{b}_i$, it will generate $(ABE.msk, ABE.PP)$ by itself. Then it will encrypt \mathbf{b}_i as \mathbf{b}_{i,F_i} to obtain ct_{i,F_i} , and will encrypt zero to obtain $ct_{i,1-F_i}$. Finally it will generate K_F by running $ABE.KeyGen$ on the function ϕ_F . This will allow generating $SCPRF.PP, K_F$ which are consistent with the distribution that \mathcal{A} receives in H_1 .

The values $SCPRF.PP, K_F$ will be sent to \mathcal{A} , and when \mathcal{A} makes PRF queries they will be forwarded to the $CPRF$ challenger, and the response forwarded back to \mathcal{A} . In addition, \mathcal{A} 's challenge will be forwarded, and the response forwarded back. When \mathcal{A} terminates and returns b' , the same b' will be returned by \mathcal{B} .

It is straightforward to see that whenever \mathcal{A} wins in H_1 , \mathcal{B} wins against $CPRF$. The claim follows. \blacksquare

Putting the two claims together, it follows that

$$1/2 + \epsilon \leq \text{Adv}_{H_0}(\mathcal{A}) \leq \text{Adv}_{H_1}(\mathcal{A}) + \text{negl}(\lambda) \leq 1/2 + \text{negl}(\lambda) ,$$

which completes the proof of the theorem. \square

References

- [Ajt96] Miklós Ajtai. Generating hard instances of lattice problems (extended abstract). In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 99–108. ACM, 1996.
- [Ale03] Michael Alekhnovich. More on average case vs approximation complexity. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*, pages 298–307. IEEE Computer Society, 2003.
- [BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 443–459, 2004.
- [BFKL93] Avrim Blum, Merrick L. Furst, Michael J. Kearns, and Richard J. Lipton. Cryptographic primitives based on hard learning problems. In *Advances in Cryptology - CRYPTO '93, 13th Annual International Cryptology Conference, Santa Barbara, California, USA, August 22-26, 1993, Proceedings*, pages 278–291, 1993.
- [BGG⁺14] Dan Boneh, Craig Gentry, Sergey Gorbunov, Shai Halevi, Valeria Nikolaenko, Gil Segev, Vinod Vaikuntanathan, and Dhinakaran Vinayagamurthy. Fully key-homomorphic encryption, arithmetic circuit ABE and compact garbled circuits. In *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, pages 533–556, 2014.
- [BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings*, pages 501–519, 2014.
- [BLMR13] Dan Boneh, Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Key homomorphic prfs and their applications. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, pages 410–428, 2013.
- [BP14] Abhishek Banerjee and Chris Peikert. New and improved key-homomorphic pseudorandom functions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 353–370. Springer, 2014.
- [BPR12] Abhishek Banerjee, Chris Peikert, and Alon Rosen. Pseudorandom functions and lattices. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 719–737. Springer, 2012.

- [BW13] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II*, pages 280–300, 2013.
- [CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *J. Cryptology*, 25(4):601–639, 2012.
- [GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. Extended abstract in FOCS 84.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pages 89–98. ACM, 2006.
- [GPV07] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(133), 2007.
- [GVW13] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Attribute-based encryption for circuits. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 545–554. ACM, 2013.
- [HKKW14] Dennis Hofheinz, Akshay Kamath, Venkata Koppula, and Brent Waters. Adaptively secure constrained pseudorandom functions. Cryptology ePrint Archive, Report 2014/720, 2014. <http://eprint.iacr.org/>.
- [KPTZ13] Aggelos Kiayias, Stavros Papadopoulos, Nikos Triandopoulos, and Thomas Zacharias. Delegatable pseudorandom functions and applications. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 669–684. ACM, 2013.
- [LMR14] Kevin Lewi, Hart William Montgomery, and Ananth Raghunathan. Improved constructions of prfs secure against related-key attacks. In Ioana Boureanu, Philippe Owesarski, and Serge Vaudenay, editors, *Applied Cryptography and Network Security - 12th International Conference, ACNS 2014, Lausanne, Switzerland, June 10-13, 2014. Proceedings*, volume 8479 of *Lecture Notes in Computer Science*, pages 44–61. Springer, 2014.
- [MM11] Daniele Micciancio and Petros Mol. Pseudorandom knapsacks and the sample complexity of LWE search-to-decision reductions. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 465–484, 2011.

- [MP12] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, pages 700–718, 2012.
- [NPR99] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed pseudo-random functions and kdcs. In Jacques Stern, editor, *Advances in Cryptology - EUROCRYPT '99, International Conference on the Theory and Application of Cryptographic Techniques, Prague, Czech Republic, May 2-6, 1999, Proceeding*, volume 1592 of *Lecture Notes in Computer Science*, pages 327–346. Springer, 1999.
- [Pei09] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: extended abstract. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 333–342, 2009.
- [Reg04] Oded Regev. Lattices in computer science - average case hardness. Lecture Notes for Class (scribe: Elad Verbin), 2004. http://www.cims.nyu.edu/~regev/teaching/lattices_fall_2004/ln/averagecase.pdf.
- [Reg05] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*, pages 84–93, 2005.
- [Sch87] Claus-Peter Schnorr. A hierarchy of polynomial time lattice basis reduction algorithms. *Theor. Comput. Sci.*, 53:201–224, 1987.
- [SW14] Amit Sahai and Brent Waters. How to use indistinguishability obfuscation: deniable encryption, and more. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 475–484. ACM, 2014.

Communication Locality in Secure Multi-party Computation

How to Run Sublinear Algorithms in a Distributed Setting

Elette Boyle¹, Shafi Goldwasser², and Stefano Tessaro^{1,*}

¹ MIT CSAIL

eboyle@mit.edu, tessaro@csail.mit.edu

² MIT CSAIL and Weizmann

shafi@theory.csail.mit.edu

Abstract. We devise multi-party computation protocols for general secure function evaluation with the property that *each party is only required to communicate with a small number of dynamically chosen parties*. More explicitly, starting with n parties connected via a complete and synchronous network, our protocol requires each party to send messages to (and process messages from) at most $\text{polylog}(n)$ other parties using $\text{polylog}(n)$ rounds. It achieves secure computation of any polynomial-time computable randomized function f under cryptographic assumptions, and tolerates up to $(\frac{1}{3} - \epsilon) \cdot n$ statically scheduled Byzantine faults.

We then focus on the particularly interesting setting in which the function to be computed is a *sublinear algorithm*: An evaluation of f depends on the inputs of at most $q = o(n)$ of the parties, where the identity of these parties can be chosen randomly and possibly adaptively. Typically, $q = \text{polylog}(n)$. While the sublinear query complexity of f makes it possible in principle to dramatically reduce the *communication complexity* of our general protocol, the challenge is to achieve this while maintaining security: in particular, while keeping the *identities* of the selected inputs completely hidden. We solve this challenge, and we provide a protocol for securely computing such sublinear f that runs in $\text{polylog}(n) + O(q)$ rounds, has each party communicating with at most $q \cdot \text{polylog}(n)$ other parties, and supports *message sizes* $\text{polylog}(n) \cdot (\ell + n)$, where ℓ is the parties' input size.

Our optimized protocols rely on a multi-signature scheme, fully homomorphic encryption (FHE), and simulation-sound adaptive NIZK arguments. However, we remark that multi-signatures and FHE are used to obtain our bounds on message size and round complexity. Assuming only standard digital signatures and public-key encryption, one can still obtain the property that each party only communicates with $\text{polylog}(n)$ other parties. We emphasize that the scheduling of faults can depend on the initial PKI setup of digital signatures and the NIZK parameters.

* This research was initiated and done in part while the authors were visiting the Isaac Newton Institute for Mathematical Sciences in Cambridge, UK.

1 Introduction

Multiparty computation (MPC) protocols for secure function evaluation (SFE) witnessed a significant body of work within the cryptography research community in the last 30 years.

These days, an emerging area of potential applications for secure MPC is to address privacy concerns in data aggregation and analysis, to match the explosive current growth of available data. Large data sets, such as medical data, transaction data, the web and web access logs, or network traffic data, are now in abundance. Much of the data is stored or made accessible in a distributed fashion. This necessitated the development of efficient distributed protocols to compute over such data. In order to address the privacy concerns associated with such protocols, cryptographic techniques such as MPC for SFE where *data items are equated with servers* can be utilized to prevent unnecessary leakage of information.

However, before MPC can be effectively used to address today's challenges, we need protocols whose efficiency and communication requirements scale practically to the modern regime of massive data. An important metric that has great effect on feasibility but has attracted surprisingly little attention thus far is the *number of other parties* that each party must communicate with during the course of the protocol. We refer to this as the communication *locality*. Indeed, if we consider a setting where potentially hundreds of thousands, or even millions of parties are participating in a computation over the internet, requiring coordination between each pair of parties will be unrealistic.

In this work, we work to optimize the communication locality for general secure function evaluation on data which is held distributively among n parties. These parties are connected via a complete synchronous communication network, of whom $(\frac{1}{3} - \epsilon)n$ may be statically scheduled, computationally bounded Byzantine faults. We do *not* assume the existence of broadcast channels.

We also focus on a particularly interesting setting in which the randomized function f to be computed is a *sublinear algorithm*: namely, a random execution of $f(x_1, \dots, x_n)$ depends on at most $q = o(n)$ of the inputs x_i . We consider both non-adaptive and adaptive sublinear algorithms, in which the identities of the selected inputs may depend on the randomness r of execution, or on both r and the values of x_i queried thus far. Sublinear algorithms play an important role in efficiently testing properties and trends when computing on large data sets. The sublinear query complexity makes it possible in principle to dramatically reduce the *amount* of information that needs to be communicated within the protocol. However, the challenge is to achieve this while maintaining security—in particular, keeping the identities of the selected inputs completely hidden.

Straightforward application of known general MPC techniques results in protocols where each party sends and receives messages from all n parties, and where the overall communication complexity is $O(n^2)$, regardless of the complexity of the function to be computed. We remark that this is obviously the case for the classical general SFE protocols (beginning with [26,14,5]) in which every party first secret shares its input among all other parties (and exchanges messages

between all n parties at the evaluation of every gate of the circuit of the function computed). Furthermore, although much progress was made in the MPC literature of the last two decades to make MPC protocols more efficient and suitable for practice, this is still the case both in works on scalable MPC [17,20,19,18] and more recent works utilizing the existence of fully homomorphic encryption schemes [35,3] for MPC. The latter achieve communication complexity that is independent of the circuit size, but not of the number of parties when broadcast channels are not available.

A recent notable exception to the need of each party to communicate with all other parties is the beautiful work of King, Saia, Sanwalani and Vee [34] on what they call scalable protocols for a relaxation of the Byzantine agreement and leader election problems. Their protocols require each honest party to send and process a $\text{polylog}(n)$ number of bits. On the down side, the protocols of [34] do not guarantee that all honest parties will achieve agreement, but only guarantee that $1 - o(1)$ fraction of the good processors reach agreement—achieving only so-called *almost everywhere agreement*. In another work of King et al [32], it is shown how using $\tilde{O}(\sqrt{n})$ communication, full Byzantine agreement can be achieved. The technique of almost-everywhere leader election of [34] will be the technical starting point of our work.

1.1 Our Results

We provide multiparty computation protocols for general secure function evaluation with communication locality that is *polylogarithmic* in the number of parties. That is, starting with n parties connected via a complete and synchronous network, we prove the following main theorem:

Theorem 1. Let f be any polynomial-time randomized functionality on n inputs. Then, for every constant $\epsilon > 0$, there exists an n -party protocol Π_f that securely computes a random evaluation of f , tolerating $t < (1/3 - \epsilon)n$ statically scheduled active corruptions, with the following complexities:

- (1) Communication locality: $\text{polylog}(n)$.
- (2) Round complexity: $\text{polylog}(n)$.
- (3) Message sizes: $O(n \cdot l \cdot \text{polylog}(n))$, where $l = |x_i|$ is the individual input size.
- (4) The protocol uses a setup consisting of $n \cdot \text{polylog}(n)$ signing keys of size $\text{polylog}(n)$, as well as a $\text{polylog}(n)$ -long additional common random string (CRS).¹

The protocol assumes a secure multisignature scheme, a fully homomorphic encryption (FHE) scheme, simulation-sound NIZK arguments, as well as pseudorandom generators.

Assuming *only* a standard signature scheme and semantically secure public-key encryption, and setup as in (4), there exists a protocol for securely computing f with $\text{polylog}(n)$ communication locality.

¹ Adversarial corruptions may be made as a function of this setup information.

Multisignatures [39,36] are digital signatures which enable the verification that a large number of signers have signed a given message, where the number of signers is not fixed in advance. The size of a multisignature is independent of the number of signers, but in order to determine their identities one must attach identifying information to the signature. Standard instantiations of such schemes exist under the bilinear computational Diffie-Hellman assumption [44,36].

The use of multisignatures rather than standard digital signatures enables us to bound the *size* of the messages sent in the protocol. Further, the use of FHE enables us to bound the *number* of messages sent, rather than depend on the time complexity of the function f to be computed and polynomially on the input size. However, we can obtain the most important feature of our complexity, the need of every party to send messages to (and process messages from) only $\text{polylog}(n)$ parties in the network, solely under the assumption that digital signatures and public-key encryption exist.

In addition, we show how to convert an arbitrary sublinear algorithm with query complexity $q = \text{polylog}(n)$ into a multi-party protocol to evaluate a randomized run of the algorithm with $\text{polylog}(n)$ communication locality and rounds, and where the *total communication complexity* sent by each party is only $O(\text{polylog}(n) \cdot (l + n))$ for $l = |x|$ an individual input size. We prove that participating in the MPC reveals no information beyond the output of the sublinear algorithm execution using a standard Ideal/Real simulation-based security definition.

For underlying query complexity q , our second main theorem is as follows:

Theorem 2. Let SLA be a sublinear algorithm which retrieves $q = q(n) = o(n)$ different inputs. Then, for all constant $\epsilon > 0$, there exists an n -party protocol Π_{SLA} that securely computes an execution of the sublinear algorithm SLA tolerating $t < (1/3 - \epsilon)n$ statically scheduled active corruptions, with the following complexities, where l is the size of the individual inputs held by the parties:

- (1) Communication locality: $q \cdot \text{polylog}(n)$.
- (2) Round complexity: $O(q) + \text{polylog}(n)$.
- (3) Message sizes: $O((l + n) \cdot \text{polylog}(n))$.
- (4) The protocol uses a setup consisting of $n \cdot \text{polylog}(n)$ signing keys of size $\text{polylog}(n)$, as well as a $\text{polylog}(n)$ -long additional CRS.

The protocol assumes a secure multisignature scheme, an FHE scheme, simulation-sound NIZK arguments, and pseudorandom generators.

Techniques. We first describe how to achieve our second result, for the case when f is a sublinear algorithm. This setting requires additional techniques in order to attain the communication complexity gains. After this, we describe the appropriate modifications required to maintain $\text{polylog}(n)$ communication locality for general functions f .

There are three main technical components to our protocol for sublinear algorithms. The first is to set up a committee structure constituted of a *supreme committee* C and n *input committees* C_1, \dots, C_n . These committees will all be of

size $\text{polylog}(n)$ and with high probability have a $2/3$ majority of honest parties. Each committee C_i will (to begin with) hold shares of the input x_i whereas the role of the supreme committee will essentially be to govern the running of the protocol. A major challenge is to ensure that all parties in the network know the identity of parties in all the committees. The starting point to address this challenge is to utilize the communication-efficient *almost-everywhere* leader election protocol of [34]. We remark that [34] achieves better total communication complexity of $\text{polylog}(n)$ bits and offers unconditional results, but only achieves an almost-everywhere agreement: there may be a $o(1)$ fraction of honest parties who will not reach agreement and, in our context, will not know the makeup of the committees. The main idea to remedy this situation is to add an iterated certification procedure using multi-signatures to the protocol of [34], while keeping the complexity of only $\text{polylog}(n)$ messages sent and processed by any honest party. In the process, however, we move from unconditional to computational security and our message sizes grow, as they will be signed by multi-signatures. Whereas the size of the multi-signatures depends only on the security parameter, the messages should indicate the identities of the signers – this is cause for the increased size of messages.

The second component is to implement a randomly chosen secret reshuffling ρ of parties' inputs within the complexity restrictions we have allotted. At the end of the shuffling, committee $C_{\rho(i)}$ will hold the input of committee C_i . Informally, this will address the major privacy issue in executing a sublinear algorithm in a distributed setting, which is to ensure that the adversary does not learn which of the n inputs are used by the algorithm. We implement the shuffling via distributed evaluation of a switching network with very good mixing properties under random switching, all under central coordination by the supreme committee. We assume that a fixed switching network over n wires is given, with depth $d = \text{polylog}(n)$, and is known to everyone.

The third component, once the inputs will be thus permuted, is to actually run the execution of the sublinear algorithm. For lack of space, let us illustrate how this is done for the sub class of non-adaptive sublinear algorithms. This is a class of algorithms that proceed in two steps:

- First, a random subset I of size q of the indices $1, \dots, n$ is selected.
- Second, an arbitrary polynomial-time algorithm is computed on inputs x_j for $j \in I$.

To run an execution of such an algorithm, the supreme committee: first selects a random and secret $q = \text{polylog}(n)$ size subset I of the inputs; and second, runs a secure function evaluation (SFE) protocol on the set of inputs in $\rho(I)$ with the assistance of parties in committees C_j for $j \in \rho(I)$. In the adaptive case, one essentially assumes queries are asked in sequence, and executes in a similar way the sublinear algorithm query after query, contacting committee $\rho(i)$ for each query i , instead of parallelizing the computation for all inputs from I . The price to pay is an additive factor q in the number of rounds of the protocol. However, note that in the common case $q = \text{polylog}(n)$, this does not affect the overall asymptotic complexity.

Now, consider the case when f is a general polynomial-time function, whose evaluation may depend on a large number of its inputs. In this case, we can skip the aforementioned shuffling procedure, and instead simply have *each party* P_i send his (encrypted) input up to the supreme committee C to run the evaluation of f . That is, each P_i gives an encryption of his input to the members of his input committee C_i , and each party in C_i sends the ciphertext up to C via a communication tree that is constructed during the process of electing committees (in Step 1). Then, the members of the supreme committee C (who collectively have the ability to decrypt ciphertexts) are able to evaluate the functionality f directly via a standard SFE.

Remarks. A few remarks are in order.

- *Flooding by faulty parties.* There is no limit (nor can there be) on how many messages are sent by faulty parties to honest parties, as is the case in the works mentioned above. To address this issue in [34,32,33,21], for example, it is (implicitly) assumed that the authenticated channels between parties can “recognize” messages from unwarranted senders which should not be processed and automatically drop them, whereas we will use a digital signature verification procedure to recognize and drop these messages which should not be processed.
- *Security definition for sublinear algorithms.* The security definition we achieve is the standard definition of secure multiparty computation (MPC). Informally, the parties will receive the output corresponding to a random execution of the sublinear algorithm but nothing else. Formally, we use the ideal/real simulation-based type definition. We note that in works of [29,23,31] on MPC for approximation algorithms for functions f , privacy is defined so as to mean that no information is revealed beyond the *exact* value of f , rather than beyond the approximate value of f computed by the protocol. One may ask for a similar privacy definition for sublinear algorithms, which are an approximation algorithm of sorts. However, this is an orthogonal concern to the one we address in this work.

1.2 Further Related Work

Work on MPC in partially connected networks, such as the recent work of Chandran, Garay and Ostrovsky [12,13], shows MPC protocols for network graphs of degree $\text{polylog}(n)$ (thus each party is connected to no more than $\text{polylog}(n)$ parties). They can only show how to achieve MPC amongst all but $o(n)$ honest parties. Indeed, in this setting it is unavoidable for some of the honest parties to be cut out from every other honest party. In contrast, in the present work, we assume that although the n parties are connected via a complete network and potentially any party can communicate with any other party, our protocols require each honest party to communicate with only at most $\text{polylog}(n)$ parties whose identity is only determined during the course of the protocol execution.

The problem of sublinear communication in MPC has also been considered in the realm of *two-party* protocols, e.g. by [40] who provide communication-preserving protocols for secure function evaluation (but which require super-polynomial computational effort), and in a recent collection of works including [28] which achieve amortized sublinear time protocols, and the work of [31] which show polylogarithmic communication for specific functions.

An interesting point of comparison to our result is the work of Halevi, Lindell and Pinkas [30]. They design computationally secure MPC protocols for n parties in which one party is singled out as a server and all other parties communicate directly with the server in sequence (in one round of communication each). However, it is easy to see that protocols in this model can only provide a limited privacy guarantee: for example, as pointed out by the authors, if the last i parties collude with the server then they can always evaluate the function on as many input settings as they wish for variable positions $n - i, n - i + 1, \dots, n$. No such limitations exist in our model.

In a recent and independent work to the current paper, King et al [21] extends [32] to show a protocol for unconditionally secure SFE for general f that requires every party to send at most $O(\frac{m}{n} + \sqrt{n})$ messages, where m is the size of a circuit representation of f . A cursory comparison to our work shows that in [21] each party sends messages to $\Omega(\sqrt{n})$ other parties.

Finally, let us point out that our approach to anonymize access patterns to parties is similar in spirit to problems arising in the context of Oblivious RAM [27], and uses similar ideas to the obfuscated secret shuffling protocols of Adida and Wilkström [2].

2 Preliminaries

We recall first the definitions of standard basic tools used throughout the paper, and then move to some important results on shuffling and our notation for sublinear algorithms.

2.1 Basic Tools

Non-interactive Zero Knowledge. We make use of a standard non-interactive zero knowledge (NIZK) argument system $(\text{Gen}, \text{Prove}, \text{Verify}, \mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{proof}}))$ with unbounded adaptive simulation soundness, as defined in [22,6,7]. That is, soundness of the argument system holds even against PPT adversaries who are given access to an oracle that produces *simulated* proofs of (potentially false) statements. For a formal definition, we refer the reader to, e.g., [22,6,7].

Theorem 1. [42] *There exists an unbounded simulation-sound NIZK proof system for any NP language L , based on trapdoor one-way permutations, with proof length $\text{poly}(|x|, |w|)$, where x is the statement and w is the witness.*

Fully Homomorphic Encryption. We make use of a fully homomorphic public-key encryption (FHE) scheme ($\text{Gen}, \text{Enc}, \text{Dec}, \text{Eval}$) as defined in, e.g., [25]. For our purposes, we require an FHE scheme with the additional property of *certifiability*. A certifiable FHE scheme is associated with a set R of “good” encryption randomness such that (repeated execution of) the Eval algorithm and the decryption algorithm Dec are correct on ciphertexts derived from those using randomness from R to encrypt. A formal definition follows.

Definition 1. For a given subset $R \subseteq \{0, 1\}^{\text{poly}(k)}$ of possible randomness values, we (recursively) define the class of R -evolved ciphertexts with respect to a public key pk to include all ciphertexts c of the form:

- $c = \text{Enc}_{\text{pk}}(m; r)$ for some m in the valid message space and randomness $r \in R$, and
- $c = \text{Eval}_{\text{pk}}((c_i)_{i \in I}, f)$ for some $\text{poly}(k)$ -size collection of R -evolved ciphertexts $(c_i)_{i \in I}$ and some poly -size circuit f .

Definition 2. A FHE scheme is said to be certifiable if there exists a subset $R \subseteq \{0, 1\}^{\text{poly}(k)}$ of possible randomness values for which the following hold.

1. $\Pr[r \in R] = 1 - \text{negl}(k)$, where the probability is over uniformly sampled $r \leftarrow \{0, 1\}^{\text{poly}(k)}$.
2. There exists an efficient algorithm \mathcal{A}_R such that $\mathcal{A}_R(r) = 1$ for $r \in R$ and 0 otherwise.
3. With overwhelming probability, Gen outputs a key pair (pk, sk) such that $\text{Dec}_{\text{sk}}(\text{Eval}_{\text{pk}}((c_i)_{1 \leq i \leq n}, f)) = f((x_i)_{1 \leq i \leq n})$ for all poly -sized circuits f and for all R -evolved ciphertexts c_1, \dots, c_n , where $x_i = \text{Dec}_{\text{sk}}(c_i)$.

Certifiable FHE schemes have been shown to exist based on the Learning with Errors assumption, together with a circular security assumption (e.g., Brakerski and Vaikuntanathan [10] and Brakerski, Gentry, and Vaikuntanathan [9]). For the readers who are familiar with these constructions, the set of “good” certifying randomness R corresponds to encrypting with sufficiently “small noise.”

Multisignatures. A multisignature scheme is a digital signature scheme with the ability to combine signatures from multiple signers on the same message into a single short object (a *multisignature*).² The first formal treatment of multisignatures was given by Micali, Ohta, and Reyzin [39].

Definition 3. A multisignature scheme is a tuple of PPT algorithms $(\text{Gen}, \text{Sign}, \text{Verify}, \text{Combine}, \text{MultiVerify})$, where syntactically $(\text{Gen}, \text{Sign}, \text{Verify})$ are as in a standard signature scheme, and $\text{Combine}, \text{MultiVerify}$ are as follows:

$\text{Combine}(\{\{\text{vk}_j\}_{j \in J_i}, \sigma_i\}_{i=1}^\ell, m)$: For disjoint $J_1, \dots, J_\ell \subseteq [n]$, takes as input a collection of signatures (or multisignatures) σ_i with respect to verification keys vk_j for $j \in J_i$, and outputs a combined multisignature, with respect to the union of verification keys.

² Note that multisignatures are a special case of *aggregate* signatures [8], which in contrast allow combining signatures from n different parties on n different messages.

MultiVerify($\{\text{vk}_i\}_{i \in I}, m, \sigma$): Verifies multisignature σ with respect to the collection of verification keys $\{\text{vk}_i\}_{i \in I}$. Outputs 0 or 1.

All algorithms satisfy the standard natural correctness properties, except with negligible probability. Moreover, the scheme is secure if for any PPT adversary \mathcal{A} , the probability that the challenger outputs 1 in the following game is negligible in the security parameter k :

Setup. The challenger samples n public key-secret key pairs, $(\text{vk}_i, \text{sk}_i) \leftarrow \text{Gen}(1^k)$ for each $i \in [n]$, and gives \mathcal{A} all verification keys $\{\text{vk}_i\}_{i \in [n]}$. \mathcal{A} selects a proper subset $M \subset [n]$ (corresponding to parties to corrupt) and receives the corresponding set of secret signing keys $\{\text{sk}_i\}_{i \in M}$.

Signing Queries. \mathcal{A} may issue multiple adaptive signature queries, of the form (m, i) . For each such query, the challenger responds with a signature $\sigma \leftarrow \text{Sign}_{\text{sk}_i}(m)$ on message m with respect to the signing key sk_i .

Output. \mathcal{A} outputs a triple $(\bar{\sigma}^*, m^*, I^*)$, where $\bar{\sigma}^*$ is an alleged forgery multisignature on message m^* with respect to a subset of verification keys $I^* \subset [n]$. The challenger outputs 1 if there exists $i \in I^* \setminus M$ such that the message m^* was not queried to the signature oracle with key sk_i , and 1 $\leftarrow \text{MultiVerify}(\{\text{vk}_i\}_{i \in I^*}, m^*, \bar{\sigma}^*)$.

The following theorem follows from a combination of the (standard) signature scheme of Waters [44] together with a transformation from this scheme to a multisignature scheme due to Lu et. al. [36].

Theorem 2. [44,36] *There exists a secure multisignature scheme with signature size $\text{poly}(k)$ (independent of message length and number of potential signers), based on the Bilinear Computational Diffie-Hellman assumption.*

Multi-party Protocols: Model and Security Definitions. We consider the setting of n parties $\mathcal{P} = \{P_1, \dots, P_n\}$ within a synchronous network who wish to jointly compute any PPT function f over their private inputs. We allow up to t statically chosen Byzantine (malicious) faults and a rushing adversary. In our protocols below, we consider $t \leq (\frac{1}{3} - \epsilon)n$ for any constant $\epsilon > 0$. We assume that every pair of parties has the ability to initiate direct communication via a point-to-point private, authenticated channel. (However, we remark that in our protocol, each (honest) party will only ever send or process information along subset of only $\text{polylog}(n)$ such channels.) We assume the existence of a public-key infrastructure, but allow the adversary's choice of corruptions to be made as a function of this public information.

The notion of security we consider is the standard simulation-based definition of secure multiparty computation (MPC), via the real/ideal world paradigm. Very loosely, we require that for any PPT adversary \mathcal{A} in a real-world execution of the protocol, there exists another PPT adversary who can simulate the output of \mathcal{A} given only access to an “ideal” world where he learns only the evaluated function output. We refer the reader to, e.g., [11] for a formal definition of (standalone) MPC security.

General secure function evaluation. The following theorem is well known and will be use throughout this paper. Let \mathbb{C} be a circuit with n inputs, and let $F_{\mathbb{C}}$ the functionality that computes the circuit.

Theorem 3. [5] *For any $t < n/3$, there exists a protocol that securely computes the functionality $F_{\mathbb{C}}$ functionality, with perfect security. The protocol proceeds in $O(|\mathbb{C}|)$ rounds, and each party sends $\text{poly}(n)$ messages of size $\text{poly}(k, n)$ each.*

Verifiable Secret Sharing. A secret sharing scheme is a protocol that allows a dealer who holds a secret input s , to share his secret among n parties such that any t parties do not gain any information about the secret s , but any set of (at least) $t + 1$ parties can reconstruct s . A *verifiable secret sharing* (VSS) scheme, introduced by Chor et al. [15], is a secret sharing scheme with the additional guarantee that after the sharing phase, a dishonest dealer is either rejected, or is committed to a single secret s , that the honest parties can later reconstruct, even if dishonest parties do not provide their correct shares.

For concreteness, we consider a class of VSS constructions that takes advantage of reconstruction and secrecy properties of low-degree polynomials [43,38]. In particular, security of such a VSS protocol **Share** is formalized as emulating the ideal functionality F_{VSS}^t for parties P_D, P_1, \dots, P_n with distinguished dealer P_D such that $F_{\text{VSS}}(q, (\emptyset, \dots, \emptyset)) = (\emptyset, (q(\alpha_1), \dots, q(\alpha_n)))$ for fixed evaluation points $\alpha_1, \dots, \alpha_n$ if $\deg(q) \leq t$, and $F_{\text{VSS}}(q, (\emptyset, \dots, \emptyset)) = (\emptyset, (\perp, \dots, \perp))$ otherwise. The party can also run a *reconstruction protocol* **Reconst** such that if honest parties input the correct shares output by the above functionality to them, then they recover the right value. The following result is well known.

Theorem 4. [5,4] *For any $t < n/3$, there exists a constant-round protocol **Share** that securely computes the F_{VSS}^t functionality, with perfect security. Each party sends $\text{poly}(n)$ messages of size $O(l \log l)$, where $l = \max\{|x|, n\}$.*

Also, we will be interested in the case where the dealer D can be any of the n parties, and he sends shares to a subset P' of the n parties of size n' (e.g., $n' = \text{polylog}(n)$), and we may not necessarily have $D \in P'$. The above functionality can be extended to this case naturally, and it is a folklore result that the protocols given by the above theorem also remain secure in this case as long as less than a fraction $1/3$ of the parties in P' are corrupted.

Broadcast. Another important functionality we need to implement is broadcast. To define, a broadcast protocol can be seen as an example of an MPC implementing a functionality F_{BC} for parties P_D, P_1, \dots, P_n with distinguished dealer P_D , defined as $F_{\text{BC}}(m, (\emptyset, \dots, \emptyset)) = (\emptyset, (m, \dots, m))$, where m is the message to be broadcast.

Theorem 5. [24] *For any $t < n/3$, there exists a constant-round protocol that securely computes the F_{BC} functionality, with perfect security. Each party sends $\text{poly}(n)$ messages of size $O(|m|)$ each.*

2.2 Random Switching Networks and Random Permutations

Our protocol will employ what we call an n -wire *switching network*, which consists of a sequence of *layers*, each layer in turn consisting of one or more swapping gates which decide to swap the values of two wires depending on a bit. Formally, given an input vector $\mathbf{x} = (x_1, \dots, x_n)$ (which we assume to be integers wlog), a swap gate operation $\text{swap}(i, j, \mathbf{x}, b)$ returns \mathbf{x}' , where if $b = 0$ then $\mathbf{x} = \mathbf{x}'$, and if $b = 1$ then we have $x'_i = x_j$, $x'_j = x_i$, and $x'_k = x_k$ for all $k \neq i, j$. A switching layer is a set $L = \{(i_1, j_1), \dots, (i_k, j_k)\}$ of pairwise-disjoint pairs of distinct indices of $[n]$. A d -depth switching network is a list $SN = (L_1, \dots, L_d)$ of switching layers. Note that for each assignment of the bits of the gates in SN , the network defines a permutation from $[n]$ to $[n]$ by inputting the vector $\mathbf{x} = (1, 2, \dots, n)$ to the network. The question we are asking is the following: If we set each bit in each swap gate uniformly and independently at random, how close to uniform is the resulting permutation? The following theorem guarantees the existence of a sufficiently shallow switching network giving rise to an almost-uniform random permutation.

Theorem 6. *For all $c > 1$, there exists an efficiently computable n -wire switching network of depth $d = O(\text{polylog}(n) \cdot \log^c(k))$ (and size $O(n \cdot d)$) such that the permutation $\hat{\pi} : [n] \rightarrow [n]$ implemented by the network when setting swaps randomly and independently has negligible statistical distance (in k) from a uniformly distributed random permutation on $[n]$.*

Proof. By Theorem 1.11 in [16], there exists such network SN of depth $d = O(\text{polylog}(n))$ where the statistical distance is of the order $O(1/n)$. Consider now the switching network SN' obtained by cascading r copies of SN . Then, when setting switching gates at random, the resulting permutation $\hat{\pi}$ equals $\hat{\pi}_1 \circ \dots \circ \hat{\pi}_r$, where $\hat{\pi}_i$ are independent permutations obtained each by setting the gates in SN uniformly at random. With π being a random permutation, a well-known property of the statistical distance $\Delta(\cdot, \cdot)$, combined with the fact permutation composition gives a group (see e.g. [37] for a proof) yields

$$\Delta(\hat{\pi}, \pi) \leq 2^{r-1} \cdot \prod_{i=1}^r \Delta(\hat{\pi}_i, \pi) \leq O\left(\left(\frac{2}{n}\right)^r\right) \leq O(2^{r(\log 2 - \log(n))}),$$

which is negligible in k for $r = \log^c(k)$. □

Note that in particular this means that each wire is connected to at most $d = O(\text{polylog}(n) \cdot \log^c(k))$ other wires via a switching gates, as each wire is part of at most one gate per layer.

2.3 Sublinear Algorithms

We consider a model where n inputs x_1, \dots, x_n are accessible to an algorithm SLA via individual queries for indices $i \in [n]$. Formally, a Q -query algorithm in the n -input model is a tuple of (randomized) polynomial time algorithms

Approved for Public Release; Distribution Unlimited.

$\text{SLA} = (\text{SLA.Sel}_1, \text{SLA.Sel}_2, \dots, \text{SLA.Sel}_Q, \text{SLA.Exec})$. During an execution with inputs (x_1, \dots, x_n) , SLA.Sel_1 takes no input and produces as output a state σ_1 and a query index $i_1 \in [n]$, and for $j = 2, \dots, n$, SLA.Sel_j takes as input a state σ_{j-1} and input $x_{i_{j-1}}$, and outputs a new state σ_j and a new query index i_j . Finally, SLA.Exec takes as input σ_Q and x_Q , and produces a final output y . We say that SLA is *sublinear* if $Q = o(n)$. We will also consider the special case of *non-adaptive* algorithms which consist without loss of generality of only two randomized algorithms $\text{SLA} = (\text{SLA.Sel}, \text{SLA.Exec})$, where SLA.Sel outputs a subset $I \subseteq [n]$ of indices of inputs to be queried, and the final output is obtained by running SLA.Exec on input $(x_i)_{i \in I}$.

Examples of sublinear algorithms, many of them non-adaptive, include algorithms for property testing such as testing sortedness of the inputs, linearity, approximate counting, and numerous graph properties, etc. Surveying this large area and the usefulness of these algorithms goes beyond the scope of this paper, and we refer the reader to the many available surveys [1].

3 Multi-party Computation for Sublinear Algorithms

We present a high-level overview geared at illustrating the techniques used within our sublinear algorithm compiler (Theorem 2), which is the more involved of our two results. For exposition, we focus on the case of non-adaptive algorithms. Given a Q -query non-adaptive sublinear algorithm SLA , we would like to evaluate it in a distributed fashion along the following lines. First, a small committee C consisting of $\text{polylog}(n)$ parties is elected, with the property that at least two thirds of its members are honest. This committee then jointly decides on a random subset of Q parties I , output by SLA.Sel , from which inputs are obtained. The parties in $C \cup I$ jointly execute a multi-party computation among themselves to produce the output of the sublinear algorithm according to the algorithm SLA.Exec , which is then broadcasted to all parties.

But things will not be as simple. Interestingly, one main challenge is very unique to the setting of sublinear algorithms: An execution of the protocol needs to hide the subset I of parties whose inputs contribute to the output! More precisely, an ideal execution of the sublinear algorithm via the functionality \mathcal{F}_{SLA} only reveals the output of the sublinear algorithm. Therefore, we need to ensure that the adversary does not learn any additional information about the composition of I from a protocol execution beyond what leaked via the final output. Our protocol will indeed hide the set I completely. This will require modifying the above naive approach considerably.

The second challenge is complexity theoretic in nature. Enforcing low complexity of our protocol when implementing the above steps, while realizing our mechanism to hide the subset I , will turn out to be a delicate balance act.

In particular, at a high level our protocol will consist of the following components:

Committee Election Phase. The n parties jointly elect a supreme committee C , as well as individual committees C_1, \dots, C_n on which they *all agree*,
 Approved for Public Release; Distribution Unlimited.

sending each at most $\text{polylog}(n)$ messages of size each $n \cdot \text{poly}(\log n, \log k)$. All committees have size $\text{polylog}(n)$ and at least a fraction $2/3$ of the parties in them are honest. As part of this process, the parties set up a communication structure that allows the supreme committee to communicate messages to all parties.

Commitment Phase. Each party P_i commits to its input so that C_i holds shares of these inputs.

Shuffling Phase. To hide the access pattern of the algorithm (i.e., which inputs are included in the computation), the committees will randomly shuffle the inputs they hold with respect to a random permutation ρ . This will happen by using a switching network with good shuffling properties. For each swap gate (i, j) in the switching network, committees C_i and C_j will swap at random the sharings they hold via a multi-party computation under a random decision taken by the supreme committee C . The supreme committee then holds a secret sharing of ρ .

Evaluation Phase. The parties in the supreme committee C sample a random query set I according to SLA.Sel via MPC and learn $\rho(I)$ only. They will then include the parties in committees C_i for $i \in \rho(I)$ in a multi-party computation to evaluate the sublinear algorithm on the inputs they hold. (Recall that C holds ρ in shared form.)

Output Phase. The supreme committee broadcasts the output of the computation to all parties, using the communication structure from the first stage.

In addition, we carefully implement sharings and multi-party computations using FHE to improve complexity, making the dependency of both the communication and round complexities linear in the input length $|x|$, rather than polynomial, and independent of the circuit sizes to implement the desired functionalities.

The following paragraphs provide a more detailed account of the techniques used within our protocol. In addition, a high-level description of the protocol procedure is given in Figure 1.

Committee Election Phase. The backbone behind this first phase is given by the construction of a *communication tree* using a technique of King et al [34]. Such tree is a sparse communication subnetwork which will ensure both the election of the supreme committee, as well as a basic form of communication between parties and the supreme committee where each party communicates only with $\text{polylog}(n)$ other parties and only $\text{polylog}(n)$ rounds of communication are required. Informally, the protocol setting up the tree assigns (possibly overlapping) subsets of parties of polylogarithmic size to the nodes of a tree with polylogarithmic height and logarithmic degree. The set of parties assigned to the root will take the role the supreme committee C . Communication from the root to the parties (or the other way round) occurs by communicating messages over paths from the root to the leaves of the tree, with an overall communication cost of $\text{polylog}(n)$ messages per party. To elect the committees C_1, \dots, C_n , we can have the supreme committee agree on the seed s of a PRF family $\mathcal{F} = \{F_s\}_s$ via a coin tossing protocol, where F_s maps elements of $[n]$ to subsets of $[n]$ of size $\text{polylog}(n)$, and send s to all parties. We then let $C_i = F_s(i)$.

However, a closer look reveals that it is only possible for the protocol building the communication tree to enforce that a vast majority of the nodes of the tree are assigned to a set of parties for which a $2/3$ majority is honest, but some nodes are unavoidably associated with too large a fraction of corrupted parties. Indeed, some parties may be connected to too many bad nodes and their communication ends up being essentially under adversarial control. As a consequence, the supreme committee is only able to correctly communicate with a $1 - o(1)$ fraction of the (honest) parties. Moreover, individual parties are not capable of determining whether the value they hold is correct or not. We refer to this situation as *almost-everywhere (ae) agreement*.

Our main contribution here is the use of cryptographic techniques to achieve *full agreement* on C and s in this stage, while maintaining $\text{polylog}(n)$ communication locality; this improves on previous work in the information-theoretic setting [32,33,21] which requires each party to talk to $O(\sqrt{n} \cdot \text{polylog}(n))$ other parties to reach agreement. We tackle these two issues in two separate ways.

1. *From ae agreement to ae certified agreement.* We first move to a stage where a large $1 - o(1)$ fraction of the parties learn the value sent by the supreme committee, together with a *proof* that the output is the one sent by the committee, whereas the remaining parties who do not know the output are also aware of this fact. We refer to this scenario as *almost-everywhere certified agreement*. Let us start with the basic idea using traditional signatures (we improve on this below using multisignatures). After having the supreme committee send a value m to all parties with almost-everywhere agreement, each party P_i receiving a value m_i will *sign* m_i with his own signing key, producing a signature σ_i . Then, P_i sends (m_i, σ_i) up the tree to the supreme committee, and each member will collect at least $n/2$ signatures on σ_i on some message m . Note that this will always be possible, as a fraction $1 - o(1) > n/2$ of the honest parties will receive the message $m_i = m$ and send a valid signature up the tree. Moreover, the adversary would need to forge signatures for honest parties in order to produce a valid certificate for a message which was not broadcast by the supreme committee.
2. *From ae certified agreement to full agreement.* We finally describe a transformation from ae certified agreement to full agreement. If a committee wants to broadcast m to all parties, the committee additionally generates a seed s for a PRF and broadcasts (m, s) in a certified way using the above transformations. Each party i receiving (m, s) with a valid certificate π forwards (m, s, π) to all parties in “his” committee $F_s(i)$. Whenever a party receives (m, s, π) with a valid certificate, it stops and outputs m . Note that no party sends more than $\text{polylog}(n)$ additional messages in this transformation. Moreover, it is not hard to see that with very high probability every honest party will be in at least one of the $F_s(i)$ for a party i who receives (m, s) correctly with a certificate, by the pseudorandomness of \mathcal{F} . Note in particular that the same seed s can be used over multiple executions of this broadcast procedure from the committee to the parties, and can be used directly to generate the committees C_1, \dots, C_n .

While we do guarantee that every party *sends* at most $\text{polylog}(n)$ messages, a problem of the above approach is the potentially high complexity of processing incoming messages if dishonest parties flood an honest party by sending too many messages. Namely, the $t = \Theta(n)$ corrupted parties can always each send (m, s) with an invalid certificate to some honest party P_i , who needs to verify all signatures in the certificate to confirm that these messages are not valid. We propose a solution based on multisignatures that alleviates this problem by making certificates only consist of an individual *aggregate* signature (instead of $\Theta(n)$), as well as of a description of the subset of parties whose signatures have been aggregated. The main idea is to have all parties initially sign the value they receive from the supreme committee with their own signing keys. However, when sending their values up the tree, parties assigned to inner nodes of the tree will aggregate valid signatures on the message which was previously sent down the tree, and keep track of which signatures have contributed.

Commitment Phase. Our instantiations of multi-party computations among subsets of parties will be based on fully homomorphic encryption (FHE). To this end, we want parties in each input committee C_i to store an FHE encryption $\text{Enc}(\text{pk}, x_i)$ of the input x_i that we want to be committing. The FHE public key pk is generated by the supreme committee (who holds secret shares of the matching secret key sk), and sent to all parties using the methods outlined above. A party i is committed to the value x_i if the honest parties in C_i all hold the *same* ciphertext encrypting x_i . This presents some challenges which we address and solve as follows:

1. First, a malicious party P_i must not be able to broadcast an invalid ciphertext to the members of the committee C_i . This is prevented by appending a simulation-sound NIZK argument π to the ciphertext c that there exists a message x and “good” randomness r such that $\text{Enc}(\text{pk}, x; r) = c$.
2. Second, for a security proof to be possible, it is well known that not only the encryption needs to be hiding and binding, but a simulator needs to be able to have some way to extract the corresponding plaintext from a valid ciphertext-proof pair (c, π) . A major issue here is that the simulated setup must be independent of the corrupted set in our model. This prevents the use of NIZK arguments of knowledge. Moreover, we can expect the FHE encryption to be secure against chosen plaintext attacks only. We will solve this by means of *double encryption*, following Sahai’s construction [41] of a CCA-secure encryption scheme from a CPA-secure one. Namely, we provide an additional encryption c_2 of x under a different public-key (for which no one needs to hold the secret key), together with an additional NIZK argument that c_1 and c_2 encrypt the same message. The ciphertext c_2 will not be necessary at any later point in time and serves only the purpose of verifying commitment validity (and permitting extraction in the proof).
3. Third, a final problem we have to face is due to rushing adversaries and the possibility of mauling commitments, in view of the use of the same public key pk for all commitments. This can be prevented in a black-box way by

letting every party P_i first (in parallel) VSS its commitment to the parties in C_i , and then in a second phase letting every committee C_i reconstruct the corresponding commitment. If the VSS protocol is perfectly secure, this ensures input-independence.

Another challenge is how to ensure that ciphertext sizes and the associated NIZK proof length are all of the order $|x| \cdot \text{poly}(k)$, instead of $\text{poly}(|x|, k)$. We achieve this by encrypting messages bit-by-bit using a bit-FHE scheme, whose ciphertexts are hence of length $\text{poly}(k)$. The corresponding NIZK proof is obtained by sequentially concatenating individual proofs (each of length $\text{poly}(k)$) for the encryptions of individual bits.

Shuffling Phase. The major privacy issue in executing a sublinear algorithm in a distributed setting is that the adversary must not learn which parties have contributed their inputs to the protocol evaluation, beyond any information that the algorithm's output itself reveals. Ideally, we would like parties to shuffle their inputs in a random (yet oblivious) fashion, so that at the end of such a protocol each party P_i holds the input of party $P_{\pi(i)}$ for a random permutation π , but such that the adversary has no information about the choice of π and for which party $\pi(i)$ he holds an input. At the same time, the supreme committee jointly holds information about the permutation π in a shared way. Unfortunately, this seems impossible to achieve: A disrupting adversary may always refuse to hold inputs for other parties. However, we can now exploit the fact that the inputs are held by *committees* C_1, \dots, C_n containing a majority of honest parties.

The actual shuffling is implemented via distributed evaluation of a switching network SN , under central coordination by the supreme committee. We assume that a switching network over n wires is given, with depth $d = \text{polylog}(n)$, and is known to everyone, and with the property given by Theorem 6: i.e., it implements a nearly uniform permutation on $[n]$ under random switching. For each swap gate (i, j) in the network, the supreme committee members jointly produce an encryption $\hat{b}_{i,j}$ of an (unknown) random bit $b_{i,j}$, indicating whether the inputs x_i and x_j are to be swapped or not when evaluating the corresponding swapping gate. The value $\hat{b}_{i,j}$ is broadcast to all parties in C_i and C_j . At this point, each party in C_i broadcasts his copy of \hat{x}_i to all parties in C_j , and each party in C_j does the same with \hat{x}_j to all parties in C_i . (Each party then, given ciphertexts from the other committee, will choose the most frequent one as the right one.) Then, each party in C_i (or C_j) will update his encryption \hat{x}_i to be an encryption of $\text{Dec}(\text{sk}, \hat{x}_j)$ or $\text{Dec}(\text{sk}, \hat{x}_i)$, depending on the value of $\hat{b}_{i,j}$, using homomorphic evaluation of the swap-or-not function. We note that this operation can be executed in parallel for all gates on the same layer, hence the swapping requires d rounds.

Evaluation Phase. Once the parties' inputs have been (obviously) shuffled, we are ready to run the sublinear algorithm. The execution is controlled by the supreme committee C . First, the members of C will run an MPC to randomly select the subset of inputs $I \subset [n]$ to be used by the algorithm. The output of

Protocol for Non-adaptive Sublinear Algorithm Evaluation (Overview)
Committee Election Phase

1. Execute *almost-everywhere* committee election protocol of [34] to generate a communication tree together with a committee C at its root (where $(1 - o(1))$ fraction of honest parties agree on C).
2. Achieve *certified* almost-everywhere agreement on C and individual committees $\{C_i\}_{i \in [n]}$ as follows. Members of C collectively sample a PRF seed s and communicate it to (almost) all parties. Each C_i is defined by $F_s(i)$. Every party signs his believed value of (C, s) and passes it up the communication tree to C , where agreeing signatures are aggregated into a single multisignature at each inner node. The message and “certificate” multisignature that contains signatures from a majority of all parties is sent back down the tree.
3. Achieve *full* agreement on $C, \{C_i\}_{i \in [n]}$ as follows. Each party P_i possessing a valid certificate π on (C, s) sends (C, s, π) to each party in $C_i := F_s(i)$. Each party P_j who does *not* have a valid certificate listens for incoming messages and adopts the first properly certified tuple. (Note steps 2-3 enable C to broadcast messages).

Commitment Phase

4. Parties in the primary committee C run the (standard) MPC protocol of [5] amongst themselves to generate keys for the FHE scheme and a second standard PKE scheme. Parties in C receive the public keys pk, pk' and a secret share of FHE key sk . They broadcast pk, pk' to all parties.
5. In parallel, each party P_i acts as dealer to VSS the following values to his input committee C_i : (1) an FHE encryption of his input $\hat{x}_i \leftarrow \text{Enc}_{\text{pk}}(x_i)$, (2) a second encryption of x_i under the standard PKE with pk' , and (3) NIZK proofs that \hat{x}_i is a valid encryption and the two ciphertexts encrypt the same value.

Shuffling Phase

6. Parties in primary committee C run an MPC to generate a random permutation ρ , expressed as a sequence of random swap bits in the switching network SN . The output is an *FHE encryption* $\hat{\rho}$ of ρ , which they broadcast to all parties.
7. The committees C_i obviously shuffle their stored input values, as follows.
For each layer L_1, \dots, L_d in the sorting network SN ,
 - Let $L_\ell = ((i_1, j_1), \dots, (i_{n/2}, j_{n/2}))$ be the swapping pairs in the current layer ℓ .
 - In *parallel*, the corresponding pairs of committees $(C_{i_1}, C_{j_1}), \dots, (C_{i_{n/2}}, C_{j_{n/2}})$ exchange their currently held input ciphertexts \hat{x}_p, \hat{x}_q (using broadcast then majority vote) and homomorphically evaluate the swap-or-not function on \hat{x}_p, \hat{x}_q , and the appropriate encrypted swap bit \hat{b} contained in $\hat{\rho}$.

Outcome: each party in committee C_i holds encryption of input $x_{\rho(i)}$.

Evaluation Phase

8. Parties in primary committee C run an MPC to execute the input selection procedure $I \leftarrow \text{SLA.Sel}$. The output of the MPC is the set of *permuted* indices $\rho(I) \subset [n]$.
9. Every party in C sends a message “Please send encrypted input ℓ ” to every party P_j in C_ℓ for which $\ell \in \rho(I)$.
10. Each party $P_j \in C_\ell$ who receives consistent messages “Please send encrypted input ℓ ” from a *majority* of the parties in C , broadcasts his currently held encrypted input $\hat{x}_{p_\ell}^j$ to all parties in C . (Recall that this allegedly corresponds to an encryption of the input x_p held by the committee $C_\ell = C_{\rho(p)}$ after the ρ -permutation shuffle).
11. The parties of C evaluate the second portion of the sublinear algorithm, SLA.Exec via an MPC. Each party of C broadcasts the resulting output *answer* to all parties.

Fig. 1. High-level overview of the protocol Π_{SLA} for secure distributed evaluation of a non-adaptive sublinear algorithm $\text{SLA} = (\text{SLA.Sel}, \text{SLA.Exec})$

Approved for Public Release; Distribution Unlimited.

the MPC will be the set of *permuted* indices $\sigma(I) := \{\sigma(i) : i \in I\}$. The corresponding committees $\{C_j : j \in \sigma(I)\}$ are invited to join in a second MPC. Each member of C_j enters the MPC with input equal to his currently held encrypted secret share (of some unknown input x_i , for which $j = \sigma(i)$). Each member of C enters the MPC with input equal to his share of the secret decryption key sk . Collectively, the members of $C \cup (\bigcup_{j \in \sigma(I)} C_j)$ run an MPC which (1) recombines the shares of sk , (2) decrypts the secret shares held by each C_j , (3) reconstructs each of the relevant inputs x_i , $i \in I$, from the corresponding set of secret shares, (4) executes the sublinear algorithm on the reconstructed inputs, and (5) outputs *only* the output value dictated by the sublinear algorithm (e.g., for many algorithms, this will simply be YES/NO).

The main challenge is making the complexity of this stage such that only $\text{poly}(\log n, \log k)$ rounds are executed, and only messages of size $|x| \cdot \text{poly}(\log k, \log n)$ will be exchanged. This will be achieved by performing most of the computations locally via FHE by the parties in the supreme committee, and by generating the randomness to be used in SLA.Sel and SLA.Exec by first agreeing on a $\text{poly}(k)$ -short seed of a PRG via coin-tossing, and then subsequently using the PRG output as the actual randomness.

Extension: Adaptive Algorithms. The above protocol can be modified to accommodate *adaptive* sublinear algorithms $\text{SLA} = (\text{SLA.Sel}_1, \dots, \text{SLA.Sel}_q, \text{SLA.Exec})$ simply by modifying the evaluation phase such that an MPC is run for each next-query SLA.Sel_j to obtain the permuted index of the next query $\rho(i_j)$. Note that without loss of generality all queries are distinct. As a result of this modification, the number of rounds unavoidably increases: Namely, we need $O(q)$ additional rounds to obtain inputs from the committees $C_{\rho(i_j)}$ one by one. However, the proof and the protocol are otherwise quite similar, and we postpone a more detailed description to the final version of this paper.

Acknowledgments. This research was initiated and done in part while the authors were visiting the Isaac Newton Institute for Mathematical Sciences in Cambridge, UK. This work was partially supported by NSF contract CCF-1018064, a Simon Investigator award, and a Visiting Fellowship of the Isaac Newton Institute for Mathematical Sciences.

This material is based on research sponsored by DARPA under agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of DARPA or the U.S. Government.

References

1. Collection of surveys on sublinear algorithms, <http://people.csail.mit.edu/ronitt/sublinear.html>
2. Adida, B., Wikström, D.: How to Shuffle in Public. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 555–574. Springer, Heidelberg (2007)
3. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., Wichs, D.: Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 483–501. Springer, Heidelberg (2012)
4. Asharov, G., Lindell, Y.: A full proof of the bgw protocol for perfectly-secure multiparty computation. IACR Cryptology ePrint Archive, 2011:136 (2011)
5. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC, pp. 1–10 (1988)
6. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications (extended abstract). In: STOC, pp. 103–112 (1988)
7. Blum, M., De Santis, A., Micali, S., Persiano, G.: Noninteractive zero-knowledge. SIAM J. Comput. 20(6), 1084–1118 (1991)
8. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Heidelberg (2003)
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: Fully homomorphic encryption without bootstrapping. In: ITCS, pp. 309–325 (2012)
10. Brakerski, Z., Vaikuntanathan, V.: Efficient fully homomorphic encryption from (standard) lwe. In: FOCS (2011)
11. Canetti, R.: Security and composition of cryptographic protocols: A tutorial. Cryptology ePrint Archive, Report 2006/465 (2006), <http://eprint.iacr.org/>
12. Chandran, N., Garay, J., Ostrovsky, R.: Improved Fault Tolerance and Secure Computation on Sparse Networks. In: Abramsky, S., Gavioille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6199, pp. 249–260. Springer, Heidelberg (2010)
13. Chandran, N., Garay, J., Ostrovsky, R.: Edge Fault Tolerance on Sparse Networks. In: Czumaj, A., Mehlhorn, K., Pitts, A., Wattenhofer, R. (eds.) ICALP 2012, Part II. LNCS, vol. 7392, pp. 452–463. Springer, Heidelberg (2012)
14. Chaum, D., Crépeau, C., Damgård, I.: Multiparty Unconditionally Secure Protocols. In: Pomerance, C. (ed.) CRYPTO 1987. LNCS, vol. 293, p. 462. Springer, Heidelberg (1988)
15. Chor, B., Goldwasser, S., Micali, S., Awerbuch, B.: Verifiable secret sharing and achieving simultaneity in the presence of faults. In: FOCS, pp. 383–395 (1985)
16. Czumaj, A., Kanarek, P., Lorys, K., Kutylowski, M.: Switching networks for generating random permutations. Manuscript (2001)
17. Damgård, I., Ishai, Y.: Scalable Secure Multiparty Computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 501–520. Springer, Heidelberg (2006)
18. Damgård, I., Ishai, Y., Krøigaard, M.: Perfectly Secure Multiparty Computation and the Computational Overhead of Cryptography. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 445–465. Springer, Heidelberg (2010)
19. Damgård, I., Ishai, Y., Krøigaard, M., Nielsen, J.B., Smith, A.: Scalable Multiparty Computation with Nearly Optimal Work and Resilience. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 241–261. Springer, Heidelberg (2008)

20. Damgård, I., Nielsen, J.B.: Scalable and Unconditionally Secure Multiparty Computation. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 572–590. Springer, Heidelberg (2007)
21. Dani, V., King, V., Movahedi, M., Saia, J.: Breaking the $o(nm)$ bit barrier: Secure multiparty computation with a static adversary. CoRR, abs/1203.0289 (2012)
22. Feige, U., Lapidot, D., Shamir, A.: Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In: FOCS, pp. 308–317 (1990)
23. Feigenbaum, J., Ishai, Y., Malkin, T., Nissim, K., Strauss, M.J., Wright, R.N.: Secure multiparty computation of approximations. ACM Transactions on Algorithms 2(3), 435–472 (2006)
24. Feldman, P., Micali, S.: Optimal algorithms for byzantine agreement. In: STOC, pp. 148–161 (1988)
25. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: STOC, pp. 169–178 (2009)
26. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game or a completeness theorem for protocols with honest majority. In: STOC, pp. 218–229 (1987)
27. Goldreich, O., Ostrovsky, R.: Software protection and simulation on oblivious rams. J. ACM 43(3), 431–473 (1996)
28. Dov Gordon, S., Katz, J., Kolesnikov, V., Malkin, T., Raykova, M., Vahlis, Y.: Secure computation with sublinear amortized work. IACR Cryptology ePrint Archive, 2011:482 (2011)
29. Halevi, S., Krauthgamer, R., Kushilevitz, E., Nissim, K.: Private approximation of np-hard functions. In: STOC, pp. 550–559 (2001)
30. Halevi, S., Lindell, Y., Pinkas, B.: Secure Computation on the Web: Computing without Simultaneous Interaction. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 132–150. Springer, Heidelberg (2011)
31. Indyk, P., Woodruff, D.P.: Polylogarithmic Private Approximations and Efficient Matching. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 245–264. Springer, Heidelberg (2006)
32. King, V., Lonargan, S., Saia, J., Trehan, A.: Load Balanced Scalable Byzantine Agreement through Quorum Building, with Full Information. In: Aguilera, M.K., Yu, H., Vaidya, N.H., Srinivasan, V., Choudhury, R.R. (eds.) ICDCN 2011. LNCS, vol. 6522, pp. 203–214. Springer, Heidelberg (2011)
33. King, V., Saia, J.: Breaking the $o(n^2)$ bit barrier: Scalable byzantine agreement with an adaptive adversary. J. ACM 58(4), 18 (2011)
34. King, V., Saia, J., Sanwalani, V., Vee, E.: Scalable leader election. In: SODA, pp. 990–999 (2006)
35. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: STOC, pp. 1219–1234 (2012)
36. Lu, S., Ostrovsky, R., Sahai, A., Shacham, H., Waters, B.: Sequential Aggregate Signatures and Multisignatures Without Random Oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 465–485. Springer, Heidelberg (2006)
37. Maurer, U.M., Pietrzak, K., Renner, R.S.: Indistinguishability Amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Heidelberg (2007)
38. McEliece, R.J., Sarwate, D.V.: On sharing secrets and reed-solomon codes. Commun. ACM 24, 583–584 (1981)
39. Micali, S., Ohta, K., Reyzin, L.: Accountable-subgroup multisignatures: extended abstract. In: ACM Conference on Computer and Communications Security, pp. 245–254 (2001)

376 E. Boyle, S. Goldwasser, and S. Tessaro

40. Naor, M., Nissim, K.: Communication preserving protocols for secure function evaluation. In: Proc. of 33rd STOC, pp. 590–599 (2001)
41. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: FOCS, pp. 543–553 (1999)
42. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust Non-interactive Zero Knowledge. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
43. Shamir, A.: How to share a secret. Communications of the ACM 22(11) (November 1979)
44. Waters, B.: Efficient Identity-Based Encryption Without Random Oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)

How to Compute in the Presence of Leakage

Shafi Goldwasser*
MIT and The Weizmann Institute of Science

Guy N. Rothblum†
Microsoft Research

Abstract

We address the following problem: how to execute any algorithm P , for an unbounded number of executions, in the presence of an adversary who observes partial information on the internal state of the computation during executions. The security guarantee is that the adversary learns nothing, beyond P 's input/output behavior.

This general problem is important for running cryptographic algorithms in the presence of side-channel attacks, as well as for running non-cryptographic algorithms, such as a proprietary search algorithm or a game, on a cloud server where parts of the execution's internals might be observed.

Our main result is a compiler, which takes as input an algorithm P and a security parameter κ , and produces a functionally equivalent algorithm P' . The running time of P' is a factor of $\text{poly}(\kappa)$ slower than P and is composed of a series of calls to $\text{poly}(\kappa)$ time computable sub-algorithms. During the executions of P' , an adversary algorithm \mathcal{A} which can choose the inputs of P' , can learn the results of adaptively chosen leakage functions—each of bounded output size $\tilde{\Omega}(\kappa)$ —on the sub-algorithms of P' and the randomness they use.

We prove that for any computationally unbounded \mathcal{A} observing the results of computationally unbounded leakage functions, will learn no more from its observations than it could given black-box access only to the input-output behavior of P . This result is unconditional and does not rely on any secure hardware components.

*Research supported by DARPA Grant FA8750-11-2-0225 and NSF Grant CCF-1018064. Email: shafi@theory.csail.mit.edu.

†Part of this research was done while the author was at Princeton University and supported by NSF Grant CCF-0832797 and by a Computing Innovation Fellowship. Email: rothblum@alum.mit.edu.

Contents

1	Introduction	1
1.1	Continual Leakage Attack Models and Prior Work	1
1.2	The New Work	3
1.3	Connections with Obfuscation	5
1.4	Other Related Work	6
2	Compiler Overview and Technical Contributions	7
2.1	Leakage-Resilient One Time Pad	8
2.2	Leakage-Resilient Compiler Overview: One-Time Secure Evaluation	10
2.2.1	Leakage Resilient <i>SafeNAND</i>	11
2.3	Leakage-Resilient Compiler Overview: Multiple Secure Evaluations	13
2.3.1	Ciphertext Banks for Secure Generation	14
2.4	Organization and Roadmap	15
3	Definitions and Preliminaries	15
3.1	Leakage Model	15
3.2	Extractors, Entropy, and Leakage-Resilient Subspaces	17
3.3	Independence up to Orthogonality	18
3.4	Secure Compiler: Definitions	20
4	Leakage-Resilient One-Time Pad (LROTP)	21
4.1	Semantic Security under Multi-Source Leakage	22
4.2	Key and Ciphertext Refreshing	22
4.3	“Safe” Homomorphic Computations	24
5	Ciphertext Banks	25
5.1	Ciphertext Bank: Interface and Security	25
5.2	Piecemeal Matrix Computations	33
5.3	Piecemeal Leakage Attacks on Matrices and Vectors	34
5.3.1	Piecemeal Leakage Resilience: One Piece	36
5.3.2	Piecemeal Leakage Resilience: Many Pieces	37
5.3.3	Piecemeal Leakage Resilience: Jointly with a Vector	41
5.4	Piecemeal Matrix Multiplication: Security	45
5.5	Ciphertext Bank Security Proofs	46
6	Safe Computations	49
6.1	Safe Computations: Interface and Security	49
6.2	Leakage-Resilient Permutation	50
6.3	Proof of <i>SimNAND</i> Security (Lemma 6.1)	54
7	Putting it Together: The Full Construction	55

1 Introduction

This work addresses the question of how to compute any program P , for an unbounded number of executions, so that an adversary who can obtain partial information on the internal states of executions of P on inputs of its choice, learns nothing about P beyond its I/O behavior.

This question is interesting for cryptographic as well as non-cryptographic algorithms. In the setting of cryptographic algorithms, the program P is usually viewed as a combination of a public algorithm with a secret key, and the secret key should be protected from side channel attacks. Stepping out of the cryptographic context, P may be a proprietary search algorithm or a novel numeric computation procedure which we want to protect, say while running on an insecure environment, say a cloud server, where its internals can be partially observed. Looking ahead, our results will not rely on any computational assumptions and thus will be applicable to non-cryptographic settings without adding any new conditions. They will hold even if one-way functions (and cryptography as we know it) do not exist.

The question of executing general computations for an unbounded (continual) number of executions, viewed largely within the context of cryptographic algorithms, has been addressed in the last few years with varying degrees of success in different adversarial settings. The crucial question seems to be how to model the partial information or *leakage* that an adversary can obtain during executions. The goal is to simultaneously capture real world attacks and achieve the right level of theoretical abstraction.

Impossibility results on obfuscation [BGI⁺01] imply inherent limitations on the leakage that can be tolerated in the continual attack model for general programs P . Even if only a single bit of leakage is output in each execution, Impagliazzo [Imp10] observes that if this bit can be computed as a function of the entire internal state of the execution, then there exist polynomial time computable functions f , for which no execution can achieve leakage resilience. Thus, to rule out this impossibility, we must put additional restriction on the leakage attack model.

1.1 Continual Leakage Attack Models and Prior Work

We discuss a few leakage attack model restrictions and corresponding results which have been considered for the question of protecting *general programs under continual leakage*.

ISW-L. The pioneering work of Ishai, Sahai, and Wagner [ISW03] first considered the question of converting general algorithms to equivalent leakage resistant algorithms. Their work views algorithms as stateful circuits (e.g. a cryptographic algorithm, whose state is the secret-key of an algorithm), and considers adversaries which can learn the value of a bounded number of wires in each execution of the circuit, whereas the values of all other wires in this execution are perfectly hidden and that all internal wire values are erased between executions. Let L be a global bound on the number of wires that can leak. Then, they show how to convert any circuit C into a new circuit C' of size $O(|C| \cdot L^2)$ which is unconditionally resilient to leakage of up to L individual wire values. In fact, their method achieves more. The new circuit C' is composed of a sequence of sub-circuits, each of size $O(L^2)$, of which the value of L arbitrary wires can leak.

CB-L. Faust, Rabin, Reyzin, Tromer and Vaikuntanathan [FRR⁺10] extended the leakage model and result of [ISW03]. They still model an algorithm as a stateful circuit, but in every execution, they let the adversary learn the result of any bounded length \mathcal{AC}^0 computable function f on the values of all the wires. Let L be a global bound on the output length of function f . under the

additional assumption that leak free hardware components exist, they show how to convert any circuit C into a new circuit C' of size $O(|C| \cdot L^2)$, which is resilient to leakage of the result of f computed on the entire set of wire values. Similarly to [ISW03], their method achieves actually more. The new circuit C' is composed of a sequence of sub-circuits, each of size $O(L^2)$, and is resilient to L bits of \mathcal{AC}^0 leakage on each of these sub-circuits.

RAM-L. the RAM model of Goldreich and Ostrovsky [GO96] considers a CPU, which loads data from fully protected memory, and runs its computations in a secure CPU. [GO96] allowed an adversary to view the access pattern to memory (and showed how to make this access pattern oblivious), but assumed that the CPU's internals and the contents of the memory are perfectly hidden.¹ This was recently extended by Ajtai [Ajt11]. He divides the execution into sub-computations. Within each sub-computation, the adversary is allowed to observe the *contents* of a constant fraction of the addresses read from memory. These are called the compromised memory accesses (or times). The contents of the un-compromised addresses, and the contents of the main memory not loaded into the CPU, are assumed to be perfectly hidden. Taking L to be a security parameter, [Ajt11] shows how to transform a program P on input size n , to a program P' which is divided into sub-computations of size $O(L)$, and is resilient to L compromised accesses in each sub-computation.

OC-L. the Micali-Reyzin [MR04] only-computation axiom assumes that there is no leakage in the absence of computation, but computation always does leak. This axiom was used in the works of Goldwasser and Rothblum [GR10] and by Juma and Vhalis [JV10], who both transform an input algorithm P (expressed as a Turing Machine or a boolean circuit) into an algorithm P' , which is divided into subcomputations. An adversary can learn the the value of *any* (adaptively chosen) polynomial time length bounded functions,² computed on each sub-computation's input and randomness. To obtain results in this model, both [GR10] and [JV10] needed to assume the existence of leak free hardware components that produce samples from a polynomial time sampleable distribution. Namely, it is assumed that there is no data leakage from the randomness generated and the computation performed inside of the device. Assuming the intractability of the DDH problem, [GR10] transform P to P' which is composed of $O(|P|)$ sub-computations, each of size $O(\text{poly}(L))$, that is resilient to leakage of length L on each sub-computation. [JV10] assume the existence of fully homomorphic encryption scheme, and get P' composed of $O(1)$ sub-computations, one of which has size $O(|P| \cdot \text{poly}(L))$. P' is resilient to leakage of length L on each sub-computation, assuming that the fully homomorphic encryption scheme cannot be broken in time $2^{O(L)}$.

The assumptions on the existence of leak-free secure hardware components make it possible, in the security proofs of [GR10] and [JV10], to argue that the view of the side channel attack in the real protocol is indistinguishable from the view output by a polynomial time simulator, which samples a very different, but computationally indistinguishable, distribution.

Finally, we mention that whereas our focus is on enabling any algorithm to run securely in the presence of continual leakage, continual leakage on *restricted computations* (e.g. [DP08, Pie09, FKPR10, BKKV10, DHLAW10, LRW11, LLW11]), and on *storage* ([DLWW11]), has been considered under various additional leakage models in a rich body of recent works. We elaborate on a few pertinent results in Section 1.4.

¹alternatively, they assume that the memory contents are encrypted, and their decryption in the CPU is perfectly hidden.

²In contrast to the \mathcal{AC}^0 restriction on f in [FRR⁺10]

1.2 The New Work

In this paper, we address the question of how to transform any algorithm P into a functionally equivalent algorithm $Eval$ which can be run for an unbounded number of executions, in the presence of leakage attacks on the internal state of the executions. Before stating our exact results, let us describe the power of our leakage-adversary, and the security guarantee to be provided

Leakage Adversary. The leakage attacks we address are in the “only computation leaks information” model of [MR04]. The algorithm $Eval$ will be composed of a sequence of calls to sub-computations. The *leakage adversary* A^λ , on input a security parameter 1^κ , can (1) specify a polynomial number of inputs to P and (2) per execution of $Eval$ on input x , request for every sub-computation of $Eval$, any λ bits of information of its choice, computed on the entire internal state of the sub-computation, including any randomness the sub-computation may generate.

We stress that we did not put any restrictions on the complexity of the leakage Adversary A^λ , and that the requested λ bits of leakage may be the result of computing a computationally unbounded function of the internal state of the sub-computation. This is in contrast to previous works that only allow the adversary to obtain polynomial-time computable functions of the execution’s internal state [GR10, JV10, BKKV10, DHLAW10, LRW11, LLW11, DLWW11].

Security Guarantee. Informally, the security guarantee that we provide will be that for any leakage adversary A^λ , whatever A^λ can compute during the execution of $Eval$, it can compute with black-box access to the algorithm P . Formally, this is proved by exhibiting a simulator which, for every leakage-adversary A^λ , given black box access to the functionality P , simulates a view which is *statistically indistinguishable* from the real view of A^λ during executions of $Eval$. The simulated view will contain the results of I/O calls to P , as well as results of applying leakage functions on the sub-computations as would be seen by A^λ . The running time of the simulator is polynomial in the running time of A^λ and the running time of the leakage functions A^λ chooses.

Informal Main Theorem. We show a compiler that takes as input a program, in the form of a circuit family $\{C_n\}$, a secret state $y \in \{0, 1\}^n$, and a security parameter κ , and produces as output a description of a uniform stateful algorithm $Eval$ such that:

1. $Eval(x) = C(y, x)$ for all inputs x .
2. The execution of $Eval(x)$ for $|x| = n$, will consist of $O(|C_n|)$ sub-computations, each of complexity (time and space) $O(poly(\kappa))$.
3. There exists a simulator Sim , a leakage bound $\lambda(\kappa) = \tilde{\Omega}(\kappa)$, and a negligible distance bound $\delta(\kappa)$, such that for every leakage-adversary $A^{\lambda(\kappa)}$ and $\kappa \in \mathbb{N}$:

$Sim^C(1^\kappa, \mathcal{A})$ is $\delta(\kappa)$ -statistically close to $view(A^\lambda)$, where $Sim^C(1^\kappa, \mathcal{A})$ denotes the output distribution of Sim , on input the description of \mathcal{A} , and with black-box access to C . $view(A^\lambda)$ is the view of the leakage adversary during a polynomial number of executions of $Eval$ on inputs of its choice. The running time of Sim is polynomial in that of \mathcal{A} and that of the leakage functions chosen by \mathcal{A} . The number of oracle calls made is always $poly(\kappa)$.

Our result holds unconditionally, without the use of computational assumptions or leak-free hardware. In Section 2 we give an overview of the construction, and highlight some of the new technical ideas of our work.

OC-L and the Leaky CPU Model. An alternative model to OC-L is that of a *leaky CPU*. We proceed with an informal description of this model. Computations are run on a RAM with two components:

1. A CPU which executes instructions from a fixed set of special universal instructions, each of size $\text{poly}(\kappa)$ for a security parameter κ .
2. A memory that stores the program, input, output, and intermediate results of the computation. The CPU fetches instructions and data and stores outputs in this memory.

The adversary model is as follows:

1. For each program instruction loaded and executed in the CPU, the adversary can learn the value of an arbitrary and adaptively chosen leakage function of bounded output length (output length $\Omega(\kappa)$ in our results). The leakage function is applied to the instruction executed in the CPU – namely, it is a function of all inputs, outputs, randomness, and intermediate wires of the CPU instruction being executed.
2. Contents of memory, when not loaded into the CPU, are hidden from the adversary.

Our result, stated in this model, provides a fixed set of CPU instructions, and a compiler which can take any polynomial time computation (say given in the form of a boolean circuit), and compile it into a program that can be run on this leaky CPU. A leakage adversary as above, who can specify inputs to the compiled program and observe its outputs, learns nothing from the execution beyond its input-out behavior.

Comparison to Prior Work. We now compare our main result to prior work on *protecting general programs under continual leakage*. See Section 1.4 for other related work.

Comparing to the work of Ishai, Sahai and Wagner [ISW03] in the ISW-L leakage model, they convert any circuit C into a new circuit C' , which is composed of $O(|C|)$ sub-circuits each of size $O(L^2)$, and allow the leakage of L arbitrary wires from each sub-circuit. Our transformation converts C into $O(|C|)$ sub-circuits, each of size $\tilde{O}(L^\omega)$, from which L arbitrary bits of information can be leaked (here ω is the exponent in the best algorithm known for matrix multiplication). These leaked bits can be the output of arbitrary computations on the wire values.

Comparing to the work of Faust *et al.* [FRR⁺10] in the CB-L model, the main differences are (i) that construction used secure hardware, whereas we do not use secure hardware, and (ii) in terms of the class of leakage tolerated, they can handle bounded-length \mathcal{AC}^0 leakage *on the entire computation* of each execution. We, on the other hand, can handle arbitrary length bounded OC-L leakage that operates separately (if adaptively) on each sub-computation.

Comparing to the work of Ajtai [Ajt11] in the RAM-L model, he divides the computation into sub-computations of size $O(L)$, and shows resilience to an adversary who see the full contents of memory loaded into CPU for L memory accesses, whereas all the other memory accesses are perfectly hidden. Translating our result to the RAM model, we divide the computation into sub-computations of size $\tilde{O}(L^\omega)$, and show resilience against an adversary that can receive L arbitrary bits of information on the entire set of memory accesses and randomness. In particular, there are no protected or hidden accesses.

Comparing to the work of Goldwasser and Rothblum [GR10] and of Juma and Vhalis [JV10] in the OC-L model, the main *qualitative* difference is that both of those prior works use computational

intractability assumptions and secure hardware. Our result, on the other hand, is unconditional and uses no secure hardware components. In terms of *quantitative* bounds, for security parameter κ , [JV10] transform a circuit of size C into a new circuit C' of size $\text{poly}(\kappa) \cdot |C|$. The new circuit C' is composed of $O(1)$ sub-circuits (one of the subcircuits is of size $\text{poly}(\kappa) \cdot |C|$). Assuming a fully-homomorphic encryption scheme that is secure against adversaries that run in time $\exp(O(L))$, their construction can withstand L bits of leakage on each sub-circuit. For example, if the FHE is secure against $\text{poly}(\kappa)$ -time adversaries, then the leakage bound is $O(\log \kappa)$. In our new construction, for leakage parameter L , there are $O(|C|)$ sub-computations (i.e. more sub-computations), each of size $\tilde{O}(L^\omega)$ (i.e. smaller), and each withstanding L bits of leakage (i.e. the amount of leakage we can tolerate, relative to the sub-computation size, is larger). The quantitative parameters of [GR10] are similar to the current work (up to polynomial factors).

Subsequent Related Work. The compiler provided in this work, and the new tools introduced in its construction, have been used in several subsequent works.

Bitansky *et al.* [BCG⁺11] use the compiler to obfuscate programs using leaky secure hardware. In a nutshell, they run each “sub-computation” on a separate leaky secure hardware component. The new challenge in that setting is providing security even when the communication channels between the components are observed and controlled by an adversary.

Boyle *et al.* [BGJK12] use the compiler to build secure MPC protocols that are resilient to corruptions of a constant fraction of the players and to leakage on each of the players (separately). The MPC should output a function of the players’ inputs computed by some circuit C . Intuitively, one can think of each player in the MPC as running one of the “sub-computations” in a compilation of C using our OC-L compiler. The additional challenges here are both adversarial monitoring/control of the communication channels and (more significantly) that the adversary may completely corrupt many of the players/sub-computations.

Using the idea of *ciphertext banks*, a technical tool introduced in this work, [Rot12] gives a compiler for \mathcal{AC}^0 leakage in the CB-L model. The new compiler removes the need for secure hardware components that was present in the work of [FRR⁺10], but its security relies on an unproven computational assumption about the power (or rather, the weakness) of \mathcal{AC}^0 circuits with pre-processing.

1.3 Connections with Obfuscation

We remark that while protecting cryptographic algorithms from side channels is an immediate application (and motivation) for this work, the question of protecting computations is interesting for non-cryptographic computations, e.g. if one-way functions do not exist. In particular, our results do not rely on cryptographic assumptions and so they would continue to hold. As a motivating example, consider a proprietary algorithm running on a cloud server, where parts of its internals might be observed.

This motivating example brings to light the fascinating connection between the problem of code obfuscation and leakage resilience for general programs. In a nut-shell, one may think of obfuscation of an algorithm as the ultimate “leakage resilient” transformation: If successful, it implies that the resulting algorithm can be “*fully leaked*” to the adversary – it is under the adversary’s complete control! Since we know that full and general obfuscation is impossible [BGI⁺01], we must relax the requirements on what we may hope to achieve when obfuscating a circuit. Leakage resilient versions of algorithms can be viewed as one such relaxation. In particular, one may view our

result as showing that although we cannot protect general algorithms if we give the adversary complete view of code which implements the algorithm (i.e obfuscation), nevertheless we can (for any algorithm) allow an adversary to have a “partial view” of the execution and only learn its black-box functionality. In our work, this “partial view” is as defined by the “only computation leaks” leakage attack model.

The recent work of Bitansky *et al.* [BCG⁺11], mentioned above, makes the connection between obfuscation and the OCL attack model even more explicit. They first strengthen the requirement of OCL attack model to allow the adversary to control the order of the execution of the sub-components (they call this DCL). They then show that any compiler that converts stateful circuits into circuits that are secure in the DML model, implies the possibility of obfuscation of any program given simple hardware components which themselves are subject to memory leakage attacks.

1.4 Other Related Work

Constructions in the OCL Leakage Model. Various constructions of *particular cryptographic primitives* [DP08, Pie09, FKPR10], such as stream ciphers and digital signatures, have been proposed in the OCL attack model and proved secure under various computational intractability assumptions. The approach in these results was to consider leakage in design time and construct new schemes which are leakage resilient, rather than a general transformation on non leakage-resilient schemes

In the context of a **bounded** number of executions, we remark that the work of Goldwasser, Kalai and Rothblum [GKR08] on one-time programs imply that any cryptographic functionality can be executed *once* in the presence of OCL attack after the initial compilation is done. There any data that is ever read or written can leak in its entirety (i.e tolerate the identity leakage function). This holds under the assumption that one-way functions exist and requires no secure hardware. The idea is that in the compilation stage, one transforms the cryptographic algorithm into a one-time program with one crucial difference. Whereas one-time programs use special hardware based memory to ensure that only certain portions of this memory cannot be read by the adversary running the one-time program, in the context of leakage the party who runs the one-time program is not an adversary but rather the honest user attempting to protect himself against OCL attacks. In the compilation stage, the honest user, stores the entire content of the special hardware based memory of [GKR08] in ordinary memory. At the execution stage, the user can be trusted to only read those memory locations necessary to run the single execution. Since an OCL attack can only view the contents of memory which are read, the execution is secure. We further observe that the follow up work of Goyal *et al.* [GIS⁺10] on one-time programs, which removes the need for the one-way function assumption, similarly implies that any cryptographic functionality can be executed *once* in the presence of OCL attacks unconditionally.

Specific Cryptographic Primitives in the Continual Memory Leakage Model. The continual memory-leakage attack model for public key encryption and digital signatures was introduced by Brakerski *et al.* [BKKV10] and Dodis *et al.* [DHLAW10]. They consider a model where an adversary can periodically compute arbitrary polynomial time functions of bounded output length L on the entire secret memory of the device. The device has an internal notion of time periods and, at the end of each period, it updates its secret key, using some fresh local randomness, maintaining the same public key throughout. As long as the rate at which the adversary can compute its leakage functions is slower than the update rate, [BKKV10, DHLAW10, LRW11, LLW11] can

construct leakage resilient public-key primitives which are still semantically secure under various intractability assumptions on problems on bi-linear groups. The continual memory leakage model is quite strong: it does not restrict the leakage functions, as in say ISW-L, to output individual wire values, or as in CB-L, to \mathcal{AC}^0 bounded functions, nor does it restrict the leakage functions to compute locally on sub-computations, as in RAM-L or OC-L. However, as pointed out by the impossibility result discussed above, this model cannot offer the kind of generality or security that we are after. In particular, the results in [BKKV10, DHLAW10, LRW11, LLW11] do not guarantee that the view the attacker obtains during the execution of a decryption algorithm is “computationally equivalent” to an attacker viewing only the I/O behavior of the decryption algorithm. For example, say an adversary’s goal in choosing its leakage requests is to compute a bit about the plain-text underlying ciphertext c . In the [BKKV10, DHLAW10] model, it will simply compute a leakage function that decrypts c , and output the requested bit. This could not be computed from the view of the I/O of the decryption algorithms decrypting ciphertexts which are unrelated to c .

Continual Leakage on a Stored Secret. A recent independent work of Dodis, Lewko, Waters, and Wichs [DLWW11], addresses the problem of how to store a value S secretly on devices that continually leak information about their internal state to an external attacker. They design a leakage resilient distributed storage method: essentially storing an encryption of S denoted $E_{sk}(S)$ on one device and storing sk on another device, for a semantically secure encryption method E which: (i) is leakage resilient under the linear assumption in prime order groups, and (ii) is “refreshable” in that the secret key sk and $E_{sk}(S)$ can be updated periodically. Their attack model is that an adversary can only leak on each device separately, and that the leakage will not “keep up” with the update of sk and $E_{sk}(S)$. One may view the assumption of leaking separately on each device as essentially a weak version of the only computation leak axiom, where locality of leakage is assumed per “device” rather than per “computation step”. We point out that storing a secret on continually leaky devices is a special case of the general results described above [ISW03, FRR⁺10, GR10, JV10] as they all must implicitly maintain the secret “state” of the input algorithm (or circuit) throughout its continual execution. The beauty of [DLWW11] is that no interaction is needed between the devices, and they can update themselves asynchronously.

We proceed to present an overview of our compiler and highlight some of our main technical contributions in Section 2 below. The full definitions, tools, and specifications of the compiler are in the subsequent sections. See the roadmap in Section 2.4.

2 Compiler Overview and Technical Contributions

The main contribution of this paper is a compiler which takes any algorithm in the form of a boolean circuit and transforms it into a functionally equivalent probabilistic stateful algorithm. A user can run this transformed secure algorithm for an unbounded (polynomial) number of executions. The security guarantee is that any computationally unbounded adversary who launches a leakage attack on the algorithm’s executions, learns nothing more than the input-output behavior.

In this section, we will give an overview of the compiler, its main components, and the technical ideas introduced. The transformed secure algorithm is executed repeatedly, on a sequence of inputs chosen by an adversary. Each execution of the transformed secure algorithm proceeds by a sequence of sub-computations, and the adversary’s view of each execution is through the results of a sequence of leakage functions (chosen adaptively and with bounded output length), applied to these sub-

computations.

The first component in our construction is a leakage-resilient one-time pad cryptosystem (LROTP), which we refer to as the *subsidiary* cryptosystem. See Section 2.1 for further details. We remark that it is important to distinguish between the leakage resilience of the secure transformed algorithm, and the leakage resilience of the subsidiary LROTP keys and ciphertexts. Whereas the LROTP scheme retains security even after direct applications of bounded output length leakage on the LROTP keys and ciphertexts (separately), the security guarantee for the transformed algorithm is that, even under a leakage attack on its execution, *there is will be no leakage at all on its internal state or secrets*. All that an adversary can learn is its input-output behavior.

Our compiler transforms a program by encrypting the bits of its description using the LROTP cryptosystem. In Section 2.2, we show how to use these encryptions to compute the program’s output on a *single* given input. This “one-time” safe evaluation is resilient to OC leakage attacks. The main new component we use is a procedure for “safe homomorphic evaluation” of LROTP-encrypted bits.

In Section 2.3 we show how to extend the one-time safe evaluation to any polynomial number of safe evaluations. This yields a compiler that is secure against continual OC leakage attacks. Here we use a new technical tool of “ciphertext banks”, which allow us to repeatedly generate secure ciphertexts even under leakage.

2.1 Leakage-Resilient One Time Pad

Our construction uses a *leakage resilient one-time pad* cryptoscheme (LROTP) as one of its main components. This simple private-key encryption scheme uses a vector $key \in \{0, 1\}^\kappa$ as its secret key, and each ciphertext is also a vector $\vec{c} \in \{0, 1\}^\kappa$. The plaintext underlying \vec{c} (under key) is the inner product: $Decrypt(key, \vec{c}) = \langle key, \vec{c} \rangle$. The scheme maintains the invariants that $key[0] = 1, \vec{c}[1] = 1$, for any key and ciphertext \vec{c} . We generate each key to be uniformly random under this invariant. To encrypt a bit b , we choose a uniformly random \vec{c} s.t. $\vec{c}[1] = 1$ and $Decrypt(key, \vec{c}) = b$.

The LROTP scheme is remarkably well suited for our goal of transforming general computations to resist leakage attacks. In particular, we highlight several properties of LROTP, specified below, that are used in our construction. See Section 4 for further details.

- **Semantic Security under Multi-Source Leakage.** Semantic security of LROTP holds against an adversary who launches leakage attacks on both a key *and* a ciphertext encrypted under that key. This might seem impossible at first glance. The reason it is facilitated is two-fold: first due to the nature of our attack model, where the adversary can never apply a leakage function to the ciphertext and the secret-key simultaneously (otherwise it could decrypt); second, the leakage from the ciphertext is of bounded length. This ensures that the adversary cannot learn enough of the ciphertext to be useful for it at a later time, when it could apply an adaptively chosen leakage function to the secret key (otherwise, again, it could decrypt).

Translating this reasoning into a proof, we show that semantic security is retained under concurrent attacks of bounded leakage $O(\kappa)$ length on key and \vec{c} . As long as leakage is of bounded length and operates separately on key and on \vec{c} , they remain (w.h.p.) high entropy sources, and are independent up to their inner product equaling the underlying plaintext. We call such sources *independent up to orthogonality*, see Definition 3.10. Since the inner product function is a two-source extractor (see Lemma 3.7), the underlying plaintext

is statistically close to uniformly random even given the leakage. Moreover, this is true even for computationally unbounded adversaries and leakage functions.

To ensure that the leakage operates separately on key and \vec{c} , we take care in our construction not to load ciphertexts and keys into working memory simultaneously. There will be one exception to this rule (see below), where a key and ciphertext will be loaded into working memory simultaneously, but this will be done only after ensuring that the ciphertext are “blinded” and contain no sensitive information.

- **Key and Ciphertext Refreshing.** We give procedures for “refreshing” LROTP keys and ciphertexts, injecting new entropy while maintaining the underlying plaintexts. We overview here the case of key refresh, ciphertext refresh is similar. The key entropy generator outputs a uniformly random $\sigma \in \{0,1\}^\kappa$ s.t. $\sigma[0] = 0$. This σ is used to inject new entropy in the key by updating $key' \leftarrow (key \oplus \sigma)$, so that key' is a uniformly random key, independent of key . σ can also be used *on its own and without knowledge of the key*, to “correlate” \vec{c} to a new ciphertext \vec{c}' s.t. $Decrypt(key', \vec{c}') = Decrypt(key, \vec{c})$. The requirement that refreshing on ciphertexts must not use the key, is due to the fact that we always want to avoid loading the ciphertext and key into memory at once (otherwise a leakage attack can decrypt and learn the plaintext). It follows that *without any leakage*, the new key or ciphertext is a uniformly random one that maintains the underlying plaintext.

In this work, key and ciphertext refreshing is used to obtain security properties *even in the presence of leakage*. One task that we will consider is permuting m key-ciphertext pairs that all have the same underlying plaintext.³ We refresh all m pairs and then permute them using a random permutation π . If there is no leakage on this refresh-and-permute procedure, then it follows that even given the m input key-ciphertext pairs, and the m refreshed-and-permuted pairs, the permutation used looks uniformly random. Furthermore, even if there is a bounded amount of leakage on the refresh-and-permute procedure, the distribution of the permutation used, given all input and output key-ciphertext pairs, will have high entropy.

The example above shows that a single application of key-ciphertext refresh can give security guarantees even in the presence of OC leakage. In particular, it maintains security of the underlying plaintext. It is natural to hope that a *large number of composed applications* of refresh to a key-ciphertext pair also maintains security of the underlying plaintext. However, after a large enough number of composed application, an OC leakage adversary can successfully reconstruct the underlying plaintext. This attack is described in Section 4.2. Intuitively, it “kicks in” once the length of the accumulated leakage is a large constant fraction of the key and ciphertext length. Our construction uses composed applications of refresh, but we take care that the accumulated leakage is never a large enough fraction of the key-ciphertext length. We show that the security properties we use are maintained under a bounded number of composed applications of refresh.

- **Homomorphic Addition.** For key and two ciphertexts \vec{c}_1, \vec{c}_2 , we can homomorphically add by computing $\vec{c}' \leftarrow (\vec{c}_1 \oplus \vec{c}_2)$. By linearity, the plaintext underlying \vec{c}' is the XOR of the plaintexts underlying \vec{c}_1 and \vec{c}_2 .

³To be precise, we will consider a related task or independently permuting m sets, each comprising 4 key-ciphertext pairs, and the ciphertexts in each set will *not* all have the same underlying plaintexts. We find the simplified question of permuting m pairs with the same underlying plaintext, as considered here, to be illuminating.

We note that the construction in [GR10] relied on several similar properties of a computationally secure public-key leakage resilient scheme: the BHHO/Naor-Segev scheme [BHHO08, NS09]. Here we achieve these properties with information theoretic security and without relying on intractability assumptions such as Decisional Diffie Hellman.

2.2 Leakage-Resilient Compiler Overview: One-Time Secure Evaluation

Here we describe the high-level structure of the compilation and evaluation algorithm for a single secure execution. In Section 2.3 we will show how to extend this framework to support any polynomial number of secure executions. We note that the high-level structure of the compilation and evaluation algorithm builds on the construction of [GR10]. The building blocks, however, are very different, as the subsidiary cryptosystem is now LROTP, and we now longer use secure hardware.

The *input* to the compiler is a *secret* input $y \in \{0, 1\}^n$, and a *public circuit* C of size $\text{poly}(n)$ that is known the adversary. The circuit takes as inputs the secret y , and also public input $x \in \{0, 1\}^n$ (which may be chosen by the adversary), and produces a single bit output.⁴ One can think of C as a universal circuit, where y describes a particular algorithm that is to be protected. Or, C can be a public cryptographic algorithm (say for producing digital signatures), and y is a secret key.

The *output* of the compiler on C and y is a probabilistic stateful evaluation algorithm $Eval$ (with a *state* which will be updated during each run of $Eval$), such that for all $x \in \{0, 1\}^n$, $C(y, x) = Eval(y, x)$. The compiler is run exactly once at the beginning of time and is not subject to leakage. See Section 3.4 for a formal definition of utility and security under leakage. In this section, we describe an initialization of $Eval$ that suffices for a single secure execution on any adversarially chosen input.

Without loss of generality, the circuit C is composed of `NAND` gates with fan-in 2 and fan-out 1, and *duplication* gates with fan-in 1 and fan-out 2. We assume a lexicographic ordering on the circuit wires, s.t. if wire k is the output wire of gate g then for any input wire i of the same gate, $i < k$. The $Eval$ algorithm keeps track of the value $v_i \in \{0, 1\}$ on each wire i of the original input circuit $C(y, x)$ in a secret-shared form: $v_i = a_i \oplus b_i$, where $a_i, b_i \in \{0, 1\}$. The invariant for every wire is that the a_i shares are *public* and known to all, including the leakage adversary, whereas b_i are *private*: they are kept encrypted by a LROTP ciphertext(s) encrypted under key_i . There is one key for each circuit wire i . For each input wire i , there is a single ciphertext \tilde{c}_i^{in} . For the output wire *output*, there is a single ciphertext \tilde{c}_{output}^{out} . For each internal wire i , an output wire for gate g and an input wire for gate h , there are *two* ciphertexts \tilde{c}_i^{out} and \tilde{c}_i^{in} (both with the same underlying plaintext b_i and the same key key_i). Intuitively, \tilde{c}_i^{out} is used in a computation corresponding to gate g (for which i is an output wire), and \tilde{c}_i^{in} is used in a computation corresponding to gate h (for which i is an input wire).

We emphasize that the adversary does not actually ever see any key or ciphertext – let alone the underlying plaintext – in their entirety. Rather, the adversary only sees the result of bounded-length leakage functions that operate separately on these keys and ciphertexts.

Initialization for One-Time Evaluation. To initialize $Eval$ for a single secure execution, we generate keys and ciphertexts for the output wires, the internal wires, and the y -input wires (initialization is performed without leakage). This is done as follows. For each bit $y[j]$ of the y -input that is carried on a wire i , we generate a key-ciphertext pair $(key_i, \tilde{c}_i^{in})$ with underlying plaintext

⁴We restrict our attention to single bit output, the case of multi-bit outputs also follows using the same ideas.

$y[j]$. The input wire's bit value v_i is thus encoded by $a_i = 0$ and $b_i = y[j]$. For each internal wire i , we choose b_i uniformly at random, and we generate a key key_i and two ciphertexts \bar{c}_i^{out} and \bar{c}_i^{in} that both have underlying plaintext b_i (under the key key_i). The internal wire's bit value v_i will be encoded by $b_i \in_R \{0, 1\}$ and $a_i = b_i \oplus v_i$ (which we have not yet computed). For the output wire $output$, we generate $(key_{output}, \bar{c}_{output}^{out})$ with underlying plaintext 0. The output bit will be encoded by $b_{output} = 0$, and a_{output} , the public share, that will equal the output value $C(y, x)$. The output wire's public share a_{output} will be computed during evaluation once the input x is specified.

This initialization suffices for a single execution (see below). Looking ahead, the main challenge for multiple execution will be securely generating the keys ciphertexts for each wire even in the presence of OC leakage. See Section 2.3.

Eval on input x . When a (non secret) input x is selected for *Eval*, we generate ciphertexts for the x -input wires. This determines the private shares (independently of the input x), and sets the stage for computing the public shares—culminating with the computation of the output wire's public share, which equals the circuit's output.

We proceed as follows. Each bit $x[j]$ of the x -input that is carried on wire i , is encoded by $a_i = x[j]$ and $b_i = 0$, where b_i is the underlying plaintext for randomly chosen (key_i, \bar{c}_i^{in}) . Given these keys and ciphertexts for the x input, and those generated in the initialization, we now have, for each circuit wire i , a key and (one or two) ciphertexts whose underlying plaintext(s) equal b_i . We also have, for each *circuit input* wire i , a public share a_i .

Eval proceeds to compute the public shares of the internal and output wires one by one, using a safe homomorphic computation procedure discussed below. The output is the public share $a_{output}^{out} = C(y, x)$. Throughout the computation, all the private b_i shares are protected from the leakage adversary. Each internal b_i looks “uniformly random” to the adversary, even under leakage. Thus, the public shares a_i of the internal wires reveal nothing about the actual values v_i on those wires. All the adversary “sees” are the input x and the output $a_o = C(y, x)$. The main remaining challenge is evaluating the public shares without exposing the private shares.

Challenge I: Leakage-Resilient “Safe NAND” Computation. We seek a procedure that, for a NAND gate takes as input the public shares for the gate's input wires, and the encrypted private shares for the gate's input wires and output wire. The output should be the correct public share of the gate's output wire. For security, we require that even under leakage, this procedure exposes nothing about the private shares of the gate's input wires and output wire (beyond the value of the output wire's public share). We also need a similar procedure for aforementioned duplication gates, but we focus here on the more challenging case of NAND. We give an overview of this procedure, which we call *SafeNAND*, in Section 2.2.1.

2.2.1 Leakage Resilient *SafeNAND*

For a NAND gate with input wires i, j and output wire k , the input to *SafeNAND* is public shares $a_i, a_j \in \{0, 1\}$, and ciphertext-key pairs $(key_i, \bar{c}_i^{in}, key_j, \bar{c}_j^{in}, key_k, \bar{c}_k^{out})$. We use $b_i, b_j, b_k \in \{0, 1\}$ to denote (respectively) the plaintext bits underlying these key-ciphertext pairs. The goal is to output

$$a_k = ((a_i \oplus b_i) \text{ NAND } (a_j \oplus b_j)) \oplus b_k$$

moreover, we want to do this using a procedure that, even under leakage, exposes nothing about (b_i, b_j, b_k) beyond the output a_k . We proceed with an overview, see Section 6 for details.

As a starting point, we first choose a fresh new $key \leftarrow KeyGen(1^\kappa)$, and compute c'_i, c'_j, c'_k whose underlying plaintexts under this new key remain b_i, b_j, b_k . This uses the key refresh property of the LROTP scheme. Once the ciphertexts are all encrypted under the same key , we can use the homomorphic addition properties of LROTP. Starting with an idea of Sanders Young and Yung [SY99], we can compute NAND by first computing a 4-tuple of encryptions:

$$C \leftarrow (\vec{c}'_k, ((a_i, 0, \dots, 0) \oplus \vec{c}'_i \oplus \vec{c}'_k), ((a_j, 0, \dots, 0) \oplus \vec{c}'_j \oplus \vec{c}'_k), ((1 \oplus a_i \oplus a_j, 0, \dots, 0) \oplus \vec{c}'_i \oplus \vec{c}'_j \oplus \vec{c}'_k))$$

Note the plaintexts underlying the 4 ciphertexts in C are:

$$(b_k, (a_i \oplus b_i \oplus b_k), (a_j \oplus b_j \oplus b_k), (1 \oplus a_i \oplus b_i \oplus a_j \oplus b_j \oplus b_k))$$

and that if $a_k = 0$, then 3 of these plaintexts will be 1, and one will be 0, whereas if $a_k = 1$, then 3 of the plaintexts will be 0 and one will be 1.

The first idea may be to simply decrypt C (using key), and compute a_k based on the number of 0's and 1's plaintext underlying C . We cannot do this, however, since the *locations* of 0's and 1's might reveal (via the adversary's leakage) information about (b_i, b_j, b_k) beyond just the value of a_k . A natural idea, then, is to *permute* the ciphertexts before decrypting. This, indeed, is what was suggested by [SY99]. Our problem, however, is that *any permutation we use might leak*. What we seek, then, is a method for randomly permuting the ciphertexts even under leakage.

Permute: Securely Permuting under Leakage. The leakage-resilient permutation procedure *Permute* that takes as input key and a 4-tuple C , consisting of 4 ciphertexts. *Permute* makes 4 copies of key , and then proceeds in iterations. The input to each iteration is two 4-tuples of keys and ciphertexts. The output from each iteration is a 4-tuple of keys and corresponding ciphertexts, whose underlying plaintexts are some permutation of those in that iteration's input. The key property is that the permutation chosen in each iteration will look “fairly random” even to a leakage adversary. As a result, the composition of these permutations over many iterations will look (statistically close to) uniformly random. The “fairly random” property of each iteration is achieved by a “duplicate and permute” step:

1. creating many copies of the input key and ciphertext 4-tuples
2. refreshing each tuple-copy using key-ciphertext refresh as in Section 2.1 (each refresh uses independent randomness)
3. permutating each tuple-copy using an independently chosen uniformly random permutation

Given (length-bounded) leakage from the above “duplicate-and-permute” step of each iteration, most of the permutations chosen will look “fairly uniform”. Finally, after the leakage from each iteration's duplicate-and-permute step has occurred, one of the tuple-copies is chosen. We will show that the permutation used for this tuple-copy will (w.h.p.) look “fairly random”, even given the leakage. The tuple-copy chosen in each iteration is then fed as input to the next iteration.

The *Permute* procedure does this for ℓ iterations. We show that the composition of all permutations used is $\exp(-\Omega(\ell))$ -statistically close to uniformly random, even given the leakage from all ℓ iterations of *Permute*. This is the high-level intuition for the security of *Permute* and *SafeNAND* (omitting many non-trivial details).

2.3 Leakage-Resilient Compiler Overview: Multiple Secure Evaluations

In this section we modify the *Init* and *Eval* procedures described in Section 2.2 to support any polynomial number of secure evaluations. The main challenge is generating secure key-ciphertext pairs for the output and the y -input wires.

Challenge II: Ciphertext Generation under Continual Leakage. We seek a procedure for repeatedly producing (key_i, \vec{c}_i) pairs. For each y -input wire i corresponding to the j -th bit of y , the underlying plaintext should be $y[j]$. For the output wire *output*, the underlying plaintext should be 0. We also seek a procedure for repeatedly producing key_i and a pair of ciphertexts $(\vec{c}_i^{out}, \vec{c}_i^{in})$ that both have the same independently random underlying plaintext $b_i \in_R \{0, 1\}$. For security, the underlying plaintexts of the keys and ciphertexts produced should be completely protected even under (repeated) leakage in all the generations.

In previous works such as [FRR⁺10, JV10, GR10], similar challenges were (roughly speaking) overcome using secure hardware to generate “fresh” encodings of leakage-resilient plaintexts from scratch in each execution.

We generate key-ciphertext pairs using *ciphertext banks*. We begin by describing this new tool and how it is for repeated secure generations with a fixed underlying plaintext bit. This is what is needed for the y -input and the output wire. We then describe how a ciphertext bank is used to generate a sequence of keys and pairs of ciphertexts (with uniformly random underlying plaintexts) for the internal wires.

A ciphertext bank is initialized once using a *BankInit*(b) procedure, where b is either 0 or 1 (there is no leakage during initialization). It can then be used, via a *BankGen* procedure, to repeatedly generate key-ciphertext pairs with underlying plaintext bit b , for an unbounded polynomial number of generations. A *BankUpdate* procedure is used between generations to inject entropy into the ciphertext bank. The intuition behind the ciphertext bank security requirement is that, even under leakage from the repeated generations, the plaintext underlying each key-ciphertext pair is protected. In particular, there are efficient simulation procedures that have arbitrary control over the plaintexts underlying the key-ciphertext pairs that the bank produces/ Leakage from the simulated calls is statistically close to leakage from the “real” ciphertext bank calls. We outline these procedures in Section 2.3.1 below. See Section 5 for details.

Using ciphertext banks, we modify the initialization and evaluation outlined in Section 2.2. In initialization, for each y -input bit $y[j]$, carried on wire i , we initialize a ciphertext bank for repeatedly generating key-ciphertext pairs with underlying plaintext $y[j]$. For the output wire we initialize a ciphertext bank for repeatedly generating key-ciphertext pairs with underlying plaintext 0. In *Eval*, we add an initial step where the ciphertext banks of each y -input wire and of the output wire are used to securely generate a key-ciphertext pair for that wire. After this first step, given an input x , *Eval* proceeds as outlined in Section 2.2.

Finally, to generate a sequence of keys and pairs of ciphertexts for the internal wires, we also provide a *BankRedraw* procedure. This procedure re-draws a new, uniformly and independently random plaintext bit, that will underly the key-ciphertext pairs produced by the bank. To generate a key and a pair of ciphertext with the same underlying plaintext we simply call *BankGen* twice: the key produced in both calls will be the same, but the ciphertexts produced will be different (albeit with the same underlying plaintext). After this pair of generations, we call *BankRedraw* to re-draw the underlying plaintext bit and then *BankUpdate* to inject new entropy. We note that it is the call to *BankUpdate* that changes the key that will be produced in future *BankGen* calls. For

security, we provide efficient simulation procedures that have arbitrary control over the plaintext bits underlying the key-ciphertext pairs that are produced. As above, leakage from simulated calls is statistically close to leakage from the “real” calls. See the overview below in Section 2.3.1, and Section 5 for further details.

We now can now repeatedly generate keys and ciphertext-pairs for internal wires even under leakage. For this, we further modify the initialization and evaluation outlined in Section 2.2. In initialization, we initialize a (single) additional ciphertext bank for the internal wires. This bank is initialized to generated key-ciphertext pairs with a uniformly random underlying plaintext bit. In *Eval*, for each internal wire we use two *BankGen* calls to this bank to generate a key and two ciphertexts. After each two such calls we use *BankRedraw* and *BankUpdate* to re-draw the underlying plaintext bit for the next wire and to inject new entropy.

This completes the high-level description of our *Init* and *Eval* procedures, the full procedures are in Section 7.

2.3.1 Ciphertext Banks for Secure Generation

The ciphertext bank state consists of an LROTP *key*, and a collection C of 2κ ciphertexts. We view C as a $\kappa \times 2\kappa$ matrix, whose columns are the ciphertexts. In the *BankInit* procedure, on input b , *key* is drawn uniformly at random, and the columns of C are drawn uniformly at random s.t the plaintext underlying each column equals b . This invariant will be maintained throughout the ciphertext bank’s operation, and we call b the bank’s underlying plaintext bit.

The *BankGen* procedure outputs *key* and a linear combination of C ’s columns. The linear combination is chosen uniformly at random s.t. it has parity 1. This guarantees that it will yield a ciphertext whose underlying plaintext is b .

The *BankUpdate* procedure injects new entropy into *key* and into C : we refresh the key using the LROTP key refresh property, and we refresh C by multiplying it with a random $2\kappa \times 2\kappa$ matrix whose columns all have parity 1. These refresh operations are performed under leakage.

The *BankRedraw* procedure chooses a uniformly random ciphertext $\vec{v} \in \{0, 1\}^\kappa$, and adds it to all the columns of C . If the inner product of *key* and \vec{v} is 0 (happens w.p. $1/2$), then the bank’s underlying plaintext bit is unchanged. If the inner product is 1 (also w.p. $1/2$), then the bank’s underlying plaintext bit is flipped.

For security, we provide a simulation procedure *SimBankGen* that can arbitrarily control the value of the plaintext bit underlying the key-ciphertext pair it generates. Here we maintain a simulated ciphertext bank, consisting of a key and a matrix, similarly to the real ciphertext bank. These are initialized, without leakage, using a *SimBankInit* procedure that draws *key* and the columns of C uniformly at random from $\{0, 1\}^\kappa$. Note that here, unlike in the real ciphertext bank, the plaintexts underlying C ’s columns are uniformly random bits (rather than a single plaintext bit b). The operation of *SimBankGen* is similar to *BankGen*, except that it uses a *biased linear combination* of C ’s columns to control the underlying plaintext it produces.

The main technical challenge and contribution here is showing that leakage from the real and simulated calls is statistically close. Note that, even for a single generation, this is non-obvious. As an (important) example, consider the rank of the matrix C : in the real view (say for $b = 0$), C ’s columns are all orthogonal to *key*, and the rank is at most $\kappa - 1$. In the simulated view, however, the rank will be κ (w.h.p). If the matrix C was loaded into memory in its entirety, then the real and simulated views would be distinguishable! Observe, however, that if only “sketches” (or “pieces”) of C are loaded into memory at any one time, where each “sketch” (or “piece”) is a collection of

$(c \cdot \kappa)$ linear combinations of C 's columns (for a small $0 < c < 1$), then it is no longer clear how a leakage adversary can compute C 's rank or distinguish a real and simulated generation (even if the adversary knows the coefficients of the linear combinations of C 's columns).

We show that: (i) *sketches of random matrices are leakage resilient*, and in particular leakage from sketches of C is statistically close in the real and simulated distributions, and (ii) how to implement *BankGen* and *SimBankGen* using subcomputations, where each sub-computation only loads a single “sketch” of C into memory. This implies security for a single generation (or a bounded number). We then extend our leakage-resilience results to show security for an unbounded (polynomial) number of generations. We view these results as our most important technical contribution.

2.4 Organization and Roadmap

Definitions, notation and preliminaries are in Section 3. This includes the definitions of secure compilers against leakage and of *independence up to orthogonality*, a central notion in many of our technical proofs. That section also includes lemmas about entropy, multi-source extractors, and leakage-resilience that will be used in the subsequent sections.

We then proceed with a full description of our construction. In Section 4 we specify the leakage-resilient one time pad scheme and its properties. We present the ciphertext bank procedures, used for secure generation of secure ciphertexts under leakage, in Section 5. The *SafeNAND* procedure for securely computing NAND gates on encrypted inputs is in Section 6. These ingredients are put together in Section 7, where we present the main construction and a proof (sketch) of its security.

3 Definitions and Preliminaries

In this section we define leakage and multi-source leakage attacks (Section 3.1) and give a brief exposition about entropy, multi-source extractors, and facts about them that will be used throughout this work (Section 3.2). We then define and discuss the notion of independence up to orthogonality (Section 3.3).

Preliminaries. For a string $x \in \Sigma^*$ (where Σ is some finite alphabet) we denote by $|x|$ the length of the string, and by x_i or $x[i]$ the i 'th symbol in the string. For a finite set S we denote by $y \in_R S$ that y drawn uniformly at random from S . We use $\Delta(D, F)$ to denote the statistical (L_1) distance between distributions D and F . For a distribution D over a finite set, we use $x \sim D$ to denote the experiment of sampling x by D , and we use $D[x]$ to denote the probability of item x by distribution D . For random variables X and Y , we use $(X|Y = y)$ or $(X|y)$ to denote the distribution of X , conditioned on Y taking value y .

3.1 Leakage Model

We follow the model and notation used in [GR10].

Leakage Attack. A leakage attack is launched on an algorithm or on a data string. In the case of a data string x , an adversary can request to see any function $\ell(x)$ whose output length is bounded by λ bits. In the case of an algorithm, the algorithm is divided into ordered sub-computations. The

adversary can request to see a bounded-length (λ bit) function of each sub-computation's input and randomness. The leakage functions are computed separately on each sub-computation, in the order in which the sub-computations occur, and can be chosen adaptively by the adversary.

Remark 3.1. *Throughout this work we focus on computationally unbounded adversaries. In particular, we do not restrict the computational complexity of the leakage functions. Moreover, without loss of generality, we consider only deterministic adversaries and leakage functions.*

Definition 3.2 (Leakage Attack $\mathcal{A}^\lambda(x)[s]$). Let s be a source: either a *data string* or a *computation*. We model a λ -bit leakage attack of adversary \mathcal{A} with input x on the source s as follows.

If s is a computation (viewed as a boolean circuit with a fixed input), it is divided into m disjoint and ordered sub-computations sub_1, \dots, sub_m , where the input to sub-computation sub_i should depend only on the output of earlier sub-computations. A λ -bit Leakage Attack on s is one in which \mathcal{A} can adaptively choose functions ℓ_1, \dots, ℓ_m , where ℓ_i takes as input the input to sub-computation i and any randomness used in that sub-computation. Each ℓ_i has output length at most λ bits. For each ℓ_i (in order), the adversary receives the output of ℓ_i on sub-computation sub_i 's input and randomness, and then chooses ℓ_{i+1} . The view of the adversary in the attack consists of the outputs to all the leakage functions.

In the case that s is a data string, we treat it as a single subcomputation.

Multi-Source Leakage Attacks. A multi-source leakage attack is one in which the adversary gets to launch concurrent leakage attacks on several sources. Each source is an algorithm or a data string. We consider both *ordered* sources, where an order is imposed on the adversary's access to the sources, and *concurrent* sources, where the leakage the leakages from each source can be interleaved arbitrarily. In both case, each leakage is computed as a function of a single source only.

Ordered Multi-Source Leakage. An *ordered* multi-source leakage attack is one in which the adversary gets to launch a leakage attack on multiple sources, where again each source is an algorithm or a data string. The attacks must occur in a specified order.

Definition 3.3 (Ordered Multi-Source Leakage Attack $\mathcal{A}(x)\{s_1^{\lambda_1}, \dots, s_k^{\lambda_k}\}$). Let s_1, \dots, s_k be leakage sources (algorithms or data strings, as in Definition 3.2). We model an *ordered* multi-source leakage attack on $\{s_1, \dots, s_k\}$ as follows. The adversary \mathcal{A} with input x runs k separate leakage attacks, one attack on each source. When attacking source s_i , the adversary can request λ_i bits of leakage. The attacks on sources s_1, \dots, s_k are run sequentially and in order, i.e. once the adversary requests leakage from s_j , it cannot get any more leakage from s_i for $i < j$.

For convenience, we drop the superscript when the source is exposed in its entirety (i.e. $\lambda_i = |s_i|$). So $\mathcal{A}(x)\{s_1^{\lambda_1}, s_2\}$ is an attack where the adversary can request λ_1 bits of leakage on s_1 , and then sees s_2 in its entirety. Finally, when the leakage bound on all k sources is identical we use a "global" leakage bound λ and denote this by $\mathcal{A}^\lambda(x)\{s_1, \dots, s_k\}$.

Concurrent Multi-Source Leakage. A *concurrent* leakage attack on multiple sources is one in which the adversary can interleave the leakages from each of the sources arbitrarily. Each leakage is still a function of a single source though. We allow additional flexibility by considering concurrent sources and ordered sources as above. Leakage from the ordered sources must obey the ordering, and the leakage from the concurrent sources can be arbitrarily interleaved with the leakage from the ordered sources.

Definition 3.4 (Multi-Source Leakage Attack $\mathcal{A}(x)[s_1^{\lambda_1}, \dots, s_k^{\lambda_k}]\{r_1^{\lambda'_1}, \dots, r_m^{\lambda'_m}\}$). Let s_1, \dots, s_k and r_1, \dots, r_m be $k + m$ leakage sources (algorithms or data strings, as in Definition 3.2). We model a *concurrent* multi-source leakage attack on $[s_1, \dots, s_k]\{r_1, \dots, r_m\}$ as follows. The adversary runs $k + m$ leakage attacks, one on each source. The attacks on each source, s_i or r_j , for a λ_i or λ'_j -bit leakage attack as in Definition 3.2. We emphasize that each λ -bit attack on a single source consists of λ adaptive choices of 1-bit leakage functions. Between different sources, the leakages can be interleaved arbitrarily and adaptively, except for each j and j' such that $j < j'$, no leakage from r_j can occur after any leakage from $r_{j'}$. There are no restrictions on the interleaving of leakages from s_i sources.

It is important that each leakage function is computed as a function of a single sub-computation in a single source (i.e. the leakages are never a function of the internal state of multiple sources). It is also important that the attacks launched by the adversary are concurrent and adaptive, and their interleaving is controlled by the adversary. For example, \mathcal{A} can request a leakage function from a sub-computation of source s_i before deciding which source to attack next, then after attacking several other sources, it can go back to source i and request a new adaptively chosen leakage attack on its next sub-computation.

As in Definition 3.3, we drop the superscript if a source s is exposed in its entirety.⁵ When the leakage from all sources is of the same length λ , we append the superscript to the adversary and drop it from the sources. If there are no ordered sources then we drop the curly braces.

3.2 Extractors, Entropy, and Leakage-Resilient Subspaces

In this section we define notions of min-entropy and two-source extractors that will be used in this work. We will then present the inner-product two-source extractor. Finally, we will state two lemmas that will be used in our proof of security: a lemma of [DRS04] about the connection between leakage and min-entropy, and a lemma of Brakerski *et al.* regarding leakage-resilient subspaces.

Definition 3.5 (Min-Entropy). For a distribution D over a domain X , its min-entropy is:

$$H_\infty(D) \triangleq \min_{x \in X} \log \Pr_{y \sim D}[y = x]$$

Definition 3.6 ($((n, m, k, \varepsilon)$ -two source strong extractor). A function $Ext : \{0, 1\}^n \times \{0, 1\}^m \rightarrow \{0, 1\}^k$ is a $((n, m, k, \varepsilon)$ -two source extractor if for every two distributions X and Y over $\{0, 1\}^n$ such that $H_\infty(X), H_\infty(Y) \geq k$ it is the case that:

$$\Pr_{y \sim Y} [\Delta(Ext(X, y), U_m) > \varepsilon] < \varepsilon$$

$$\Pr_{x \sim X} [\Delta(Ext(x, Y), U_m) > \varepsilon] < \varepsilon$$

Chor and Goldreich [CG88] showed that the inner-product function over any field is a two-source extractor. See also the excellent exposition of Rao [Rao07]. The claims made in those works imply the lemma below (they make more general statements).

⁵we use this only for the ordered sources, concurrent sources exposed in their entirety are w.l.o.g. given to the adversary as part of its input.

Lemma 3.7 (Inner-Product Extractor [CG88]). For $\kappa \in \mathbb{N}$ and $\vec{x}, \vec{y} \in \mathbb{GF}[2]^\kappa$ define

$$\text{Ext}(\vec{x}, \vec{y}) = \langle \vec{x}, \vec{y} \rangle$$

For any $\kappa \in \mathbb{N}$, the function $\text{Ext}(x, y)$ is a $(\kappa, 1, 0.51\kappa, 2^{-\Omega(\kappa)})$ -two source strong extractor.

Finally, we will use the fact that bounded-length multi-source (or rather two-source) leakage attacks on high-entropy sources X and Y , leave an adversary with a view that is statistically close to one in which each of the sources comes from a high-entropy distribution. This follows from a result of Dodis *et al.* [DRS04].

Lemma 3.8 (Residual Entropy after Leakage [DRS04]). Let X and Y be two sources with min-entropy at least k . Then for any leakage adversary \mathcal{A} , taking $w = \mathcal{A}^\lambda[X, Y]$, consider the conditional distributions $X' = (X|w)$ and $Y' = (Y|w)$, which are just X and Y conditioned on leakage w . For any $\delta > 0$, with probability at least $1 - \delta$ over the choice of w , $H_\infty(X'), H_\infty(Y') \geq k - \lambda - \log(1/\delta)$.

3.3 Independence up to Orthogonality

Definition 3.9 (Independent up to Orthogonality (IuO) Distribution on Vectors). Let \mathcal{D} be a distribution over pairs $(\vec{x}, \vec{y}) \in \{0, 1\}^\kappa \times \{0, 1\}^\kappa$. We say that \mathcal{D} is IuO w.r.t. $\vec{v} \in \{0, 1\}^\kappa$ and $b \in \{0, 1\}$, if there exist distributions \mathcal{X} and \mathcal{Y} , both over $\{0, 1\}^\kappa$, s.t. \mathcal{D} is obtained by sampling $\vec{x} \sim \mathcal{X}$ and then sampling $\vec{y} \sim \mathcal{Y}$, conditioned on $\langle \vec{x} + \vec{v}, \vec{y} \rangle = b$. We call \mathcal{X} and \mathcal{Y} the *underlying distributions* of \mathcal{D} , and denote this by $\mathcal{D} = \mathcal{X} \perp_{(\vec{v}, b)} \mathcal{Y}$.

When $\vec{v} = \vec{0}$ we will sometimes simply say that \mathcal{D} is IuO with orthogonality b , and denote this by $\mathcal{D} = \mathcal{X} \perp_b \mathcal{Y}$.

We also consider the *independently drawn variant* of \mathcal{D} which is obtained by independently sampling $\vec{x} \sim \mathcal{X}$ and $\vec{y} \sim \mathcal{Y}$. We denote the independently drawn variant by \mathcal{D}^\times or $\mathcal{X} \times \mathcal{Y}$.

Definition 3.10 (Independent up to Orthogonality (IuO) Distribution on Matrices). Generalizing Definition 3.10, for an integer $m \geq 1$, let \mathcal{D} be a distribution over pairs $(X, Y) \in \{0, 1\}^{m \times \kappa} \times \{0, 1\}^{m \times \kappa}$. We say that \mathcal{D} is IuO w.r.t. $V \in \{0, 1\}^{m \times \kappa}$ and $\vec{b} \in \{0, 1\}^m$ if there exist distributions \mathcal{X} and \mathcal{Y} , both over $\{0, 1\}^{m \times \kappa}$, s.t. \mathcal{D} is obtained by sampling $X \sim \mathcal{X}$ and then (independently) sampling $Y \sim \mathcal{Y}$ conditioned on $\forall i \in [m], \langle X[i] + V[i], Y[i] \rangle = \vec{b}[i]$. As in Definition 3.10, we call \mathcal{X} and \mathcal{Y} the *underlying distributions* of \mathcal{D} , and denote this by $\mathcal{D} = \mathcal{X} \perp_{(V, \vec{b})} \mathcal{Y}$.

When V is the all-zeros matrix, we will sometimes simply say that \mathcal{D} is IuO with orthogonality \vec{b} , and denote this by $\mathcal{D} = \mathcal{X} \perp_{\vec{b}} \mathcal{Y}$.

We also consider the *independently drawn variant* of \mathcal{D} which is obtained by independently sampling $X \sim \mathcal{X}$ and $Y \sim \mathcal{Y}$. We denote the independently drawn variant by \mathcal{D}^\times or $\mathcal{X} \times \mathcal{Y}$.

Finally, for a distribution \mathcal{D} over pairs $(\vec{x}, Y) \in \{0, 1\}^\kappa \times \{0, 1\}^{m\kappa}$, we say that \mathcal{D} is IuO (with parameters as above), if \mathcal{D}' , in which we replace \vec{x} with a matrix X whose columns are m (identical) copies of \vec{x} is IuO (as above). We emphasize that the copies of \vec{x} are all identical and completely dependant.

One important property of IuO distributions, which we will use repeatedly, is that they are indistinguishable from their independently drawn variant under multi-source leakage (as long as they have sufficient entropy).

Lemma 3.11. *Let \mathcal{D} be an IuO distribution over pairs $(X, Y) \in S_X \times S_Y$, with underlying distributions \mathcal{X} and \mathcal{Y} . Suppose that $S_X = \{0, 1\}^{m_X \cdot \kappa}$ and $S_Y = \{0, 1\}^{m_Y \cdot \kappa}$ for m_X and m_Y s.t. $1 \leq m_X \leq m_Y \leq 10$. Suppose also that $H_\infty(\mathcal{D}) \geq (m_X + m_Y - 0.3) \cdot \kappa$. Then for any (computationally unbounded) multi-source leakage adversary \mathcal{A} , and leakage bound $\lambda \leq 0.1\kappa$, taking the following two distributions:*

$$\begin{aligned} \text{Real} &= \left(\mathcal{A}^\lambda[X, Y] \right)_{(X, Y) \sim \mathcal{D}} \\ \text{Simulated} &= \left(\mathcal{A}^\lambda[X, Y] \right)_{(X, Y) \sim \mathcal{D}^\times} \end{aligned}$$

it is the case that $\Delta(\text{Real}, \text{Simulated}) = \exp(-\Omega(\kappa))$.

Moreover, for any w in the support of Real : (i) we can derive from \mathcal{X} a conditional underlying distribution $\mathcal{X}(w)$, and from \mathcal{Y} a conditional underlying distribution $\mathcal{Y}(w)$. In particular, note that \mathcal{D} is not needed for computing these conditional underlying distributions. Taking $\mathcal{D}(w) = (\mathcal{D}|w)$ to be the conditional distribution of \mathcal{D} , given leakage w , then $\mathcal{D}(w)$ is IuO, with underlying distributions $\mathcal{X}(w)$ and $\mathcal{Y}(w)$.

Before proving the lemma, we consider a simple application to multi-source leakage from two strings. In *Real* the strings are uniformly random with inner product 0, and in *Simulated* they are independently uniformly random. By Lemma 3.11, the leakage in both cases is statistically close. The distribution of the strings in *Real*, given the leakage, is IuO, and each of its underlying distributions can be computed (separately) given the leakage (and that the original underlying distribution were uniformly random).

Proof of Lemma 3.11. Take $w = \mathcal{A}^\lambda[X, Y]$. Since the leakage operates separately on X and on Y , there exist two sets $S_X(w) \subseteq S_X$ and $S_Y(w) \subseteq S_Y$, s.t.:

$$w = \mathcal{A}^\lambda[X, Y] \Leftrightarrow (X, Y) \in S_X(w) \times S_Y(w)$$

We take $\mathcal{X}(w)$ to be \mathcal{X} conditioned on $X \in S_X(w)$, and $\mathcal{Y}(w)$ to be \mathcal{Y} conditioned on $Y \in S_Y(w)$. Let $\mathcal{D}(w) = (\mathcal{D}|w)$ be the distribution \mathcal{D} conditioned on leakage w . By the above, $\mathcal{D}(w)$ is \mathcal{D} conditioned on $(X, Y) \in S_X(w) \times S_Y(w)$. Thus, $\mathcal{D}(w)$ is also IuO, with underlying distributions $\mathcal{X}(w)$ and $\mathcal{Y}(w)$ and the same orthogonality as \mathcal{D} .

Finally, to show that *Real* and *Simulated* are statistically close, let $\beta(w)$ denote the distance of the inner product $\langle X + V, Y \rangle_{X \sim \mathcal{X}(w), Y \sim \mathcal{Y}(w)}$ from uniform.

Claim 3.12. *For any $w \in \text{Support}(\text{Real})$:*

$$1 - O(\beta(w)) \leq \frac{\text{Simulated}[w]}{\text{Real}[w]} \leq 1 + O(\beta(w))$$

Proof. Observe that:

$$\begin{aligned} \text{Simulated}[w] &= \Pr_{X \sim \mathcal{X}, Y \sim \mathcal{Y}}[(X, Y) \in S_X(w) \times S_Y(w)] \\ \text{Real}[w] &= \Pr_{(X, Y) \sim \mathcal{D}}[(X, Y) \in S_X(w) \times S_Y(w)] \\ &= \Pr_{X \sim \mathcal{X}, Y \sim \mathcal{Y}'(X)}[(X, Y) \in S_X(w) \times S_Y(w)] \end{aligned}$$

where $\mathcal{Y}'(X)$ is \mathcal{Y} conditioned on $\langle X + V, Y \rangle = \vec{b}$.

The claim follows because:

$$1 - O(\beta(w) \cdot 2^{m_Y}) \leq \frac{\Pr_{X \sim \mathcal{X}, Y \sim \mathcal{Y}}[\langle X + V, Y \rangle = \vec{b}]}{\Pr_{X \sim \mathcal{X}, Y \sim \mathcal{Y}}[\langle X + V, Y \rangle = \vec{b} | (X, Y) \in S_X(w) \times S_Y(w)]} \leq 1 + O(\beta(w) \cdot 2^{m_Y})$$

■

Claim 3.13. *With all but $\exp(-\Omega(\kappa))$ probability over $w \sim \text{Real}$, $\beta(w) = \exp(-\Omega(\kappa))$.*

Proof. By Lemma 3.8, with all but δ probability over $w \sim \text{Real}$, we have that $H_\infty(\mathcal{X}(w)) + H_\infty(\mathcal{Y}(w)) \geq (m_X + m_Y - 0.45) \cdot \kappa$. When this is the case, by Lemma 3.7 we have $\beta(w) = \exp(-\Omega(\kappa))$. ■

By Claim 3.12 and 3.13 we conclude that $\Delta(\text{Real}, \text{Simulated}) = \exp(-\Omega(\kappa))$. ■

3.4 Secure Compiler: Definitions

We now present formal definitions for a secure compiler against continuous and computationally unbounded leakage. We view the input to the compiler as a circuit C that is known to all parties and takes inputs x and y . The input y is fixed, whereas the input x is chosen by the user. The user can adaptively choose inputs x_1, x_2, \dots and the functionality requirement is that on each input x_i the user receives $C(y, x_i)$. The secrecy requirement is that even for a computationally unbounded adversary who chooses the inputs (say polynomially many inputs in the security parameter), even giving the adversary access (repeatedly) to a leakage attack on the secure transformed computation, the adversary learns nothing more than the circuit's outputs. In particular, the adversary should not learn y .⁶

We divide a compiler into parts: the first part, the *initialization* occurs only once at the beginning of time. This procedure depends only on the circuit C being compiled and the private input y . We assume that during this phase there is no leakage. The second part is the *evaluation*. This occurs whenever the user wants to evaluate the circuit $C(y, \cdot)$ on an input x . In this part the user specifies an input x , the corresponding output $C(y, x)$ is computed under leakage.

Definition 3.14 ($(\lambda(\cdot), \delta(\kappa))$ Continuous Leakage Secure Compiler). We say that a compiler $(\text{Init}, \text{Eval})$ for a circuit family $\{C_n(y, x)\}_{n \in \mathbb{N}}$, where C_n operates on two n -bit inputs, is $(\lambda(\cdot), \delta(\kappa))$ -secure under continuous leakage, if for every integer $n, \kappa \in \mathbb{N}$, and every $y \in \{0, 1\}^n$, the following hold:

- Initialization: $\text{Init}(1^\kappa, C_n, y)$ runs in time $\text{poly}(\kappa, n)$ and outputs an initial state state_0
- Evaluation: for every integer $t \leq \text{poly}(\kappa)$, the evaluation procedure is run on the previous state state_{t-1} and an input $x_t \in \{0, 1\}^n$. We require that for every $x_t \in \{0, 1\}^n$, when we run:

$$(\text{out}_t, \text{state}_t) \leftarrow \text{Eval}(\text{state}_{t-1}, x_t)$$

with all but negligible probability over the coins of Init and the t invocations of Eval , $\text{out}_t = C_n(y, x_t)$.

⁶Unless, of course, y can be computed from the outputs of the circuit on the inputs the adversary chose.

- $(\lambda(\kappa), \delta(\kappa))$ -Continuous Leakage Security: There exists a simulator Sim , s.t. for every (computationally unbounded) leakage adversary \mathcal{A} , the view $Real_{\mathcal{A}}$ of \mathcal{A} when adaptively choosing $T = \text{poly}(\kappa)$ inputs (x_1, x_2, \dots, x_T) while running a continuous leakage attack on the sequence $(Eval(state_0, x_1), \dots, Eval(state_{T-1}, x_T))$, with adaptively and adversarially chosen x_t 's, is $(\delta(\kappa))$ -statistically close to the view $Simulated_{\mathcal{A}}$ generated by Sim , which only gets the description of the adversary and the input-output pairs $((x_1, C(y, x_1)), \dots, (x_T, C(y, x_T)))$.

Formally, the adversary repeatedly and adaptively, in iterations $t \leftarrow 1, \dots, T$, chooses an input x_t and launches a $\lambda(\kappa)$ -bit leakage attack on $Eval(state_{t-1}, x_t)$ (see Definition 3.2). $Real_{\mathcal{A},t}$ is the view of the adversary in iteration t , including the input x_t , the output o_t , and the (aggregated) leakage w_t from the t -th iteration. The complete view of the adversary is

$$Real_{\mathcal{A}} = (Real_{\mathcal{A},1}, \dots, Real_{\mathcal{A},T})$$

a random variable over the coins of the adversary, of $Init$ and of $Eval$ (in all of its iterations).

The simulator's view is generated by running the adversary with *simulated* leakage attacks. The simulator includes $SimInit$ and $SimEval$ procedures. The initial state is generated using $SimInit$. Then, in each iteration t the simulator gets the input x_t chosen by the adversary and the circuit output $C(y, x_t)$. It generates simulated leakage w_t . It is important that the simulator sees nothing of the internal workings of the evaluation procedure. We compute:

$$state_0 \leftarrow SimInit(1^\kappa, C_n)$$

$$x_t \leftarrow \mathcal{A}(Simulated_{\mathcal{A},1}, \dots, Simulated_{\mathcal{A},t-1})$$

$$(state_t, Simulated_{\mathcal{A},t}) \leftarrow SimEval(state_{t-1}, x_t, C(y, x_t), \mathcal{A}, Simulated_{\mathcal{A},1}, \dots, Simulated_{\mathcal{A},t-1})$$

where $Sim_{\mathcal{A},t}$ is a random variable over the coins of the adversary when choosing the next input and of the simulator. The complete view of the simulator is

$$Simulated_{\mathcal{A}} = (Simulated_{\mathcal{A},1}, \dots, Simulated_{\mathcal{A},T})$$

We require that the two views $Real_{\mathcal{A}}$ and $Simulated_{\mathcal{A}}$ are $(\exp(-\Omega(\kappa)))$ -statistically close.

We note that modeling the leakage attacks requires dividing the *Eval* procedure into sub-computations. In our constructions, the size of these sub-computations will always be $O(\kappa^\omega)$, where ω is the exponent in the running time of an algorithm for matrix multiplication.

4 Leakage-Resilient One-Time Pad (LROTP)

In this section we present the *leakage resilient one-time pad* cryptoscheme, a main component of our construction. See the overview in Section 2.1. Here we specify the scheme and its properties that will be used in the main construction.

Leakage-Resilient One-Time Pad (LROTP) Cryptosystem ($KeyGen, Encrypt, Decrypt$)

- $KeyGen(1^\kappa)$: output a uniformly random $key \in \{0, 1\}^\kappa$ s.t. $key[0] = 1$
- $CipherGen(1^\kappa)$: output a uniformly random $\vec{c} \in \{0, 1\}^\kappa$ s.t. $\vec{c}[1] = 1$.
- $Encrypt(key, b \in \{0, 1\})$: output a uniformly random $\vec{c} \in \{0, 1\}^\kappa$ s.t. $\vec{c}[1] = 1$ and $\langle key, \vec{c} \rangle = b$
- $Decrypt(key, \vec{c})$: output $\langle key, \vec{c} \rangle$

Figure 1: Leakage-Resilient One-Time Pad (LROTP) Cryptosystem

4.1 Semantic Security under Multi-Source Leakage

Definition 4.1 (Semantic Security Under $\lambda(\cdot)$ -Multi-Source Leakage). An encryption scheme $(KeyGen, Encrypt, Decrypt)$ is semantically secure under computationally unbounded multi-source leakage attacks if for every (unbounded) adversary \mathcal{A} , when we run the game below, the adversary's advantage in winning (over $1/2$) is $\exp(-\Omega(\kappa))$:

1. The game chooses key $key \leftarrow KeyGen(1^\kappa)$, chooses uniformly at random a bit $b \in_R \{0, 1\}$, and generates a ciphertext $\vec{c} \leftarrow Encrypt(key, b)$.
2. The adversary launches a leakage attack on key and \vec{c} , and outputs a “guess” b' :

$$b' \leftarrow \mathcal{A}^{\lambda(\kappa)}(1^\kappa)[key, \vec{c}]$$

the adversary wins if $b' = b$.

Lemma 4.2. *The LROTP cryptosystem, as defined in Figure 1, is semantically secure in the presence of multi-source leakage with leakage bound $\lambda(\kappa) = \kappa/3$.*

Proof. The proof follows directly from Lemma 3.11. ■

4.2 Key and Ciphertext Refreshing

As discussed in the introduction, the LROTP scheme supports procedures for injecting new entropy into a key or a ciphertext. This is done using *entropy generators* $KeyEntGen$ and $CipherEntGen$. The values these procedures produce can be used to refresh a key or ciphertext using $KeyRefresh$ or $CipherRefresh$ (respectively). Key entropy σ can also be used, *without knowledge of key*, to correlate a ciphertext \vec{c} so that the plaintext underlying the correlated ciphertext \vec{c}' under $key' \leftarrow KeyRefresh(key, \sigma)$, is equal to the plaintext underlying \vec{c} under key . This is done using the $CipherCorrelate$ procedure. A similar $KeyCorrelate$ procedure for correlating keys using ciphertext entropy. These procedures are all in Figure 2 below.

We proceed with a discussion of the security properties of the refreshing procedures, and their limitation. For a key-ciphertext pair (key, \vec{c}) , a *refresh operation* on the pair injects new entropy into the key and the ciphertext, while maintaining the underlying plaintext, as follows:

1. $\sigma \leftarrow KeyEntGen(1^\kappa)$
2. $key' \leftarrow KeyRefresh(key, \sigma)$

LROTP key and ciphertext refresh

- $KeyEntGen(1^\kappa)$: output a uniformly random $\sigma \in \{0, 1\}^\kappa$ s.t. $\sigma[0] = 0$
- $KeyRefresh(key, \sigma)$: output $key \oplus \sigma$
- $CipherCorrelate(\vec{c}, \sigma)$: modify $\vec{c}[0] \leftarrow \vec{c}[0] \oplus \langle \vec{c}, \sigma \rangle$, and then output \vec{c}
- $CipherEntGen(1^\kappa)$: output a uniformly random $\tau \in \{0, 1\}^\kappa$ s.t. $\tau[1] = 0$
- $CipherRefresh(\vec{c}, \tau)$: output $\vec{c} \oplus \tau$
- $KeyCorrelate(key, \tau)$: modify $key[1] \leftarrow key[1] \oplus \langle key, \tau \rangle$, and then output key

Figure 2: LROTP key and ciphertext refresh Cryptosystem

3. $\vec{c}' \leftarrow CipherCorrelate(\vec{c}, \sigma)$
4. $\pi \leftarrow CipherEntGen(1^\kappa)$
5. $\vec{c}'' \leftarrow CipherRefresh(\vec{c}', \pi)$
6. $key'' \leftarrow KeyCorrelate(key', \pi)$

The output of the refresh operation is (key'', \vec{c}'') . We treat each step of the key-refresh as a sub-computation, and so the leakage operates separately on the keys and on the ciphertexts.

Security Properties. The security properties of the refreshing procedures are, first, that a key-ciphertext pair can be refreshed without ever loading the key and ciphertext into memory at the same time, i.e. while operating separately on the key and on the ciphertext. We will use this to argue that an OC leakage adversary learns nothing about the plaintext bit underlying a pair that is being refreshed (as long as the total amount of leakage is bounded). The second property we use is that *without any leakage*, a the refreshed pair is a uniformly random key-ciphertext pair with the same underlying plaintext bit.

We use these properties to prove security of the *Permute* procedure which is used in *SafeNAND* (see Sections 2.2.1 and 6.2). *Permute* proceeds in iterations. In each iteration, we refresh a tuple of key-ciphertext pairs and then permute them using a random permutation. The property of the refresh procedure that we will use is that *without any leakage*, even given both the input and the output of a single iteration of *Permute*, *nothing is leaked about the permutation chosen* (beyond what can be gleaned from the underlying plaintexts). This will then be used to argue that, even under a bounded amount of leakage from each iteration, the permutation chosen in each iteration of *Permute* has (w.h.p.) high entropy. This is later used to prove the security of *SafeNAND*.

Refresh Forever? It is natural to ask whether key-ciphertext refreshing maintains security of the underlying plaintext under OC leakage for an unbounded polynomial number of refreshings. If so, we could hope to do away with the (significantly more complicated) ciphertext banks, replacing the ciphertexts generated by each bank with a sequence of ciphertexts generated using repeated refresh calls. Unfortunately, there is an OC attack that exposes the plaintext underlying a key-ciphertext pair that is refreshed too many times. The attack is outlined below.

We consider a sequence of refresh operations, where the output of the i -th refresh is used as input for the $(i + 1)$ -th refresh. During the first refresh, an OC adversary leaks the inner product (i.e. the product) of the first bit of the output key and the first bit of the output ciphertext. This requires only one bit of leakage from each. In the second refresh, the adversary will learn the inner product of the first *two* bits of the output key and the output ciphertext. To do so, let (key_1, \vec{c}_1) be the inputs to the second refresh. The adversary leaks the second bits of key_2 during *KeyRefresh*, and of \vec{c}_2 during *CipherRefresh*. It also keeps track of the *change* in inner product of the *first* bit of $key'_1 = (key_1 + \sigma)$ and of $\vec{c}'_1 = \text{CipherCorrelate}(\vec{c}_1, \sigma)$ using a single bit of leakage: The change (w.r.t. the inner product of key_1 and \vec{c}_1) is just a function of σ and \vec{c}_1 , which are loaded into memory during *CipherCorrelate*. Similarly, the adversary can keep track of the subsequent change to the inner product of the first bits of $key_2 = \text{KeyCorrelate}(key'_1, \pi)$ and $\vec{c}_2 = \vec{c}'_1 \oplus \pi$, using a single bit of leakage from *KeyCorrelate*. Putting these pieces together, the adversary learns the inner product of the first two bits of key_2 and \vec{c}_2 . More generally, after the i -th refresh call, the key point is that if the adversary knows the inner product of the first i bits of the input key and ciphertext, it can track the change in this inner product for the output key and cipher. Tracking the change requires only two bits of OC leakage. The adversary uses two additional bits of OC leakage to expand its knowledge to the inner product of the first $(i + 1)$ bits.

Continuing the above attack for κ refresh calls, the adversary learns the inner product of the key and ciphertext obtained, i.e. the underlying plaintext is exposed. Note that this used only $O(1)$ bits of leakage from each sub-computation. If ℓ bits of leakage from each sub-computation were allowed, then the underlying plaintext would be exposed after $O(\kappa/\ell)$ refresh calls. When using refresh, we will take care that the total leakage accumulated from a sequence of refresh calls to a key-ciphertext pair will be well under κ bits. Since refresh operates separately on keys and ciphertexts, the semantic security of LROTP in the presence of multi-source leakage will guarantee that the underlying plaintext is hidden.

4.3 “Safe” Homomorphic Computations

The LROTP cryptoscheme supports homomorphic computation on ciphertexts as follows:

Homomorphic Addition. For key and two ciphertexts \vec{c}_1, \vec{c}_2 , we can homomorphically add by computing $\vec{c}' \leftarrow (\vec{c}_1 \oplus \vec{c}_2)$. By linearity, the plaintext underlying \vec{c}' is the XOR of the plaintexts underlying \vec{c}_1 and \vec{c}_2 .

Homomorphic NAND. LROTP supports safe computation of a masked NAND functionality. This functionality takes three input key-ciphertext pairs, and outputs the NAND of the first two underlying plaintexts, XORed with the third underlying plaintext. Moreover, this can be performed via the *SafeNAND* procedure, which guarantees that even an OC leakage attacker who gets leakage on the computation, learns nothing about the input plaintexts beyond the procedure’s output. See Sections 2.2.1 and 6 for details.

We note that this can be extended to “standard” homomorphic computation of NAND, where the input is two key-ciphertext pairs, and the output is a “blinded” key-ciphertext pair whose underlying plaintext is the NAND of the plaintexts underlying the inputs. The details are omitted (this second property follows from the security of *SafeNAND*, but is not used in our construction).

5 Ciphertext Banks

In this section we present the procedures for maintaining, utilizing, and simulating banks of secure ciphertexts. We use these to create fresh secure ciphertexts under leakage attacks. The security property we want is that, even though the generation of new ciphertexts is done under leakage, a simulator can create an indistinguishable simulated view with complete and arbitrary control over these ciphertexts' underlying plaintexts. See Section 2.3.1 for an overview.

This section is organized as follows. In Section 5.1 we describe the ciphertext bank procedures, and those of the simulator, and state the security properties that will be used in the main construction (the proofs follow in subsequent sections). These procedures make use of secure procedures for piecemeal matrix multiplication and for refreshing collections of ciphertexts, which are in section Section 5.2. In Section 5.3 we define piecemeal attacks on matrices and prove that random matrices are resilient to piecemeal leakage. In Section 5.4 we state and prove security properties of piecemeal matrix multiplication. Finally, we use these claims to prove the ciphertext bank's security. We conclude with proofs of the ciphertext bank's security in Section 5.5.

5.1 Ciphertext Bank: Interface and Security

We present a full description of the ciphertext bank procedures and simulator. Recall that (as in Section 4), keys and ciphertexts are vectors in $\{0,1\}^\kappa$, and the decryption of ciphertext \vec{c} under key is the inner product $b = \langle key, \vec{c} \rangle$. We call b the plaintext underlying ciphertext \vec{c} .

Ciphertext Bank Procedures. The ciphertext bank is used to generate fresh ciphertext-key pairs. The bank is initialized (without leakage) using a *BankInit* procedure that takes as input a bit $b \in \{0,1\}$. It can then be accessed (repeatedly) using a *BankGen* procedure, which produces a key-ciphertext pair whose underlying plaintext is b . Between generations, the bank's internal state is updated using a *BankUpdate* Procedure. Leakage from a sequence of *BankGen* and *BankUpdate* calls can be simulated. The simulator has arbitrary control over the plaintext bits underlying the generated ciphertexts. Simulated leakage is statistically close to leakage from the real calls.

In addition, we provide a *BankRedraw* procedure. The *BankRedraw* procedure re-draws a uniformly random plaintext bit that will underly ciphertexts produced by the bank. The redrawn plaintext bit looks uniformly random even in the presence of leakage on the *BankRedraw* procedure (and on all ciphertext generations).

These functionalities are implemented as follows. The ciphertext bank consists of key and a collection C of 2κ ciphertexts. We view C as a $\kappa \times 2\kappa$ matrix, whose columns are the ciphertexts.

In the *BankInit* procedure, on input b , the keys is drawn uniformly at random, and the columns of C are drawn uniformly at random s.t their inner product with key is b . This invariant will be maintained throughout the ciphertext bank's operation. We sometimes refer to b as the ciphertext bank's *underlying plaintext bit*.

The *BankGen* procedure outputs a linear combination of C 's columns. The linear combination is chosen uniformly at random s.t. it has parity 1. This guarantees that it will yield a ciphertext whose underlying plaintext is b . The linear combination is taken using a secure "piecemeal" matrix-vector multiplication procedure *PiecemealMM*.

The *BankUpdate* procedure injects new entropy into key and into C . We refresh the key using a ("piecemeal") key refresh procedure *PiecemealRefresh*. We refresh C by multiplying it with a

random matrix whose columns all have parity 1. Matrix multiplication is again performed securely using *PiecemealMM*.

The *BankRedraw* procedure adds a uniformly random vector in $\{0, 1\}^\kappa$ to each column of C (here key is left unchanged). With probability $1/2$, the vector has inner product 1 with key , and the underlying plaintext bit is flipped. Otherwise, the underlying plaintext bit is unchanged. Adding the vector to each column of the matrix is performed using a secure *PiecemealAdd* procedure.

The full ciphertext bank procedures are in Figure 3. The piecemeal matrix multiplication, addition, and key refresh procedures are below in Section 5.2.

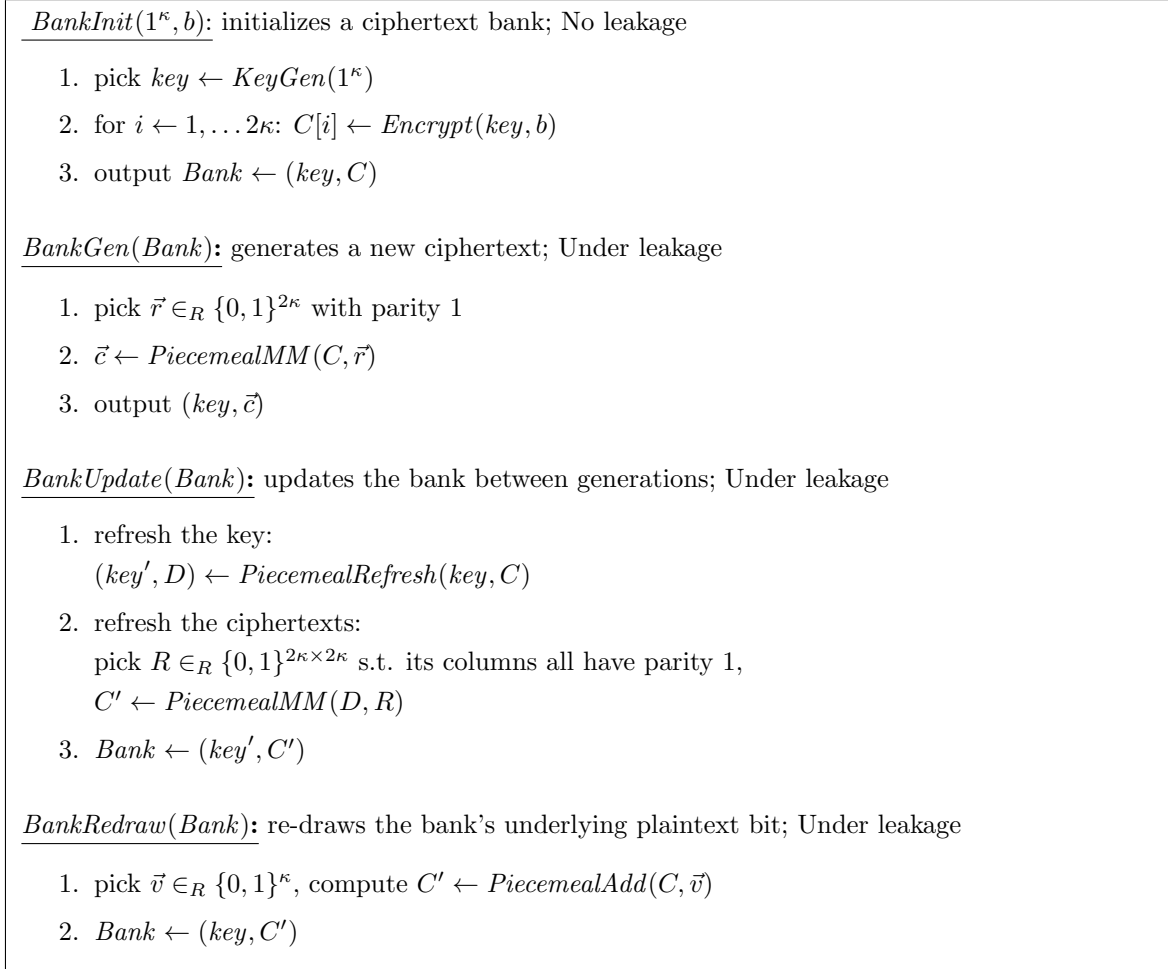


Figure 3: Ciphertext Bank

Simulated Ciphertext Bank. Next, we provide a simulator for simulating the ciphertext bank procedure, while arbitrarily controlling the plaintext bits underlying the ciphertexts that are produced. Towards this end, we maintain a simulated ciphertext bank, consisting of a key and a matrix, similarly to the real ciphertext bank. These are initialized, without leakage, in a *SimBankInit* procedure that draws key and the columns of C uniformly at random from $\{0, 1\}^\kappa$. Note that here, unlike in the real ciphertext bank, the plaintexts underlying C 's columns are independent and uni-

formly random bits (rather than all 0 or all 1). The simulator also keeps track of the plaintexts bits underlying the columns of C , storing them in a vector $\vec{x} \in \{0, 1\}^{2\kappa}$.

Calls to $BankGen$ are simulated using $SimBankGen$. This procedure operates similarly to $BankGen$, except that it uses a biased linear combination of C 's columns to control the plaintext underlying its output ciphertext. We also provide $SimBankUpdate$ and $SimBankRedraw$ procedures. These operate similarly to $BankUpdate$ and $BankRedraw$, except that they keep track of changes to the vector \vec{x} of plaintext bits underlying C . The simulation procedures are in Figure 4.

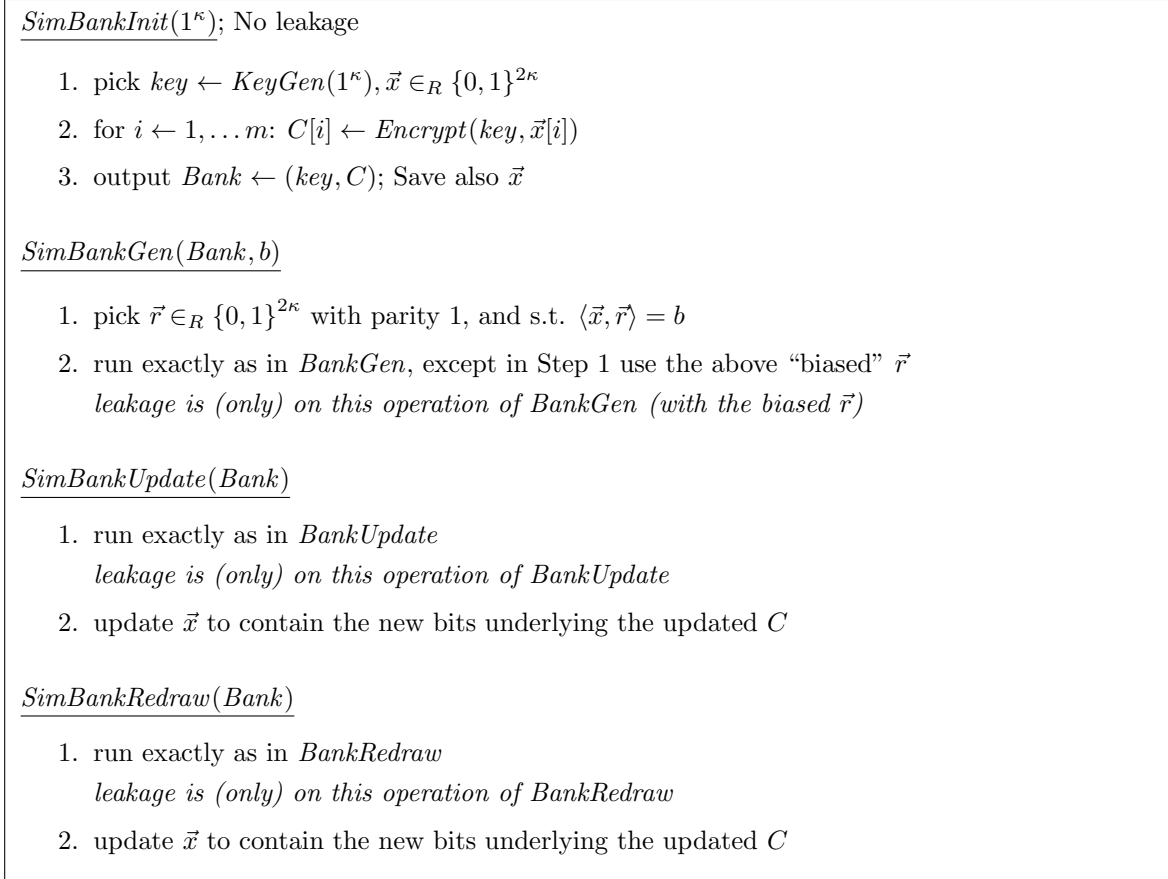


Figure 4: Simulated Ciphertext Bank

Ciphertext Bank Security. We show several security properties of the ciphertext bank. In all of these security properties, we consider sequences of ciphertext bank generations, real or simulated. A *sequence of real generations* starts with a call to $BankInit$ to initialize the ciphertext bank. This is followed by a sequence of ciphertext generations, each performed via a call to $BankGen$, and followed by an update call to $BankUpdate$. A *sequence of simulated generations* is similar, except that initialization is performed using $SimBankInit$, each generation is performed by specifying an underlying plaintext bit b and then calling $SimBankGen$, and each update is performed using $SimBankUpdate$.

We also consider sequences of random generations of ciphertext-pairs. A *sequence of real random generations* begins with an initialization call to $BankInit$ with a uniformly random bit value.

This is followed by a sequence of generations as follows. For each item in the sequence, we begin by generating a key and *two* ciphertexts, \bar{c}^α and \bar{c}^β (both with the same underlying plaintext bit). Next, we call *BankRedraw* to redraw the bank’s underlying plaintext bit. Lastly, we update the bank using *BankUpdate*. This is done repeatedly, yielding a sequence of keys and pairs of ciphertexts, where the plaintext bit underlying each ciphertext pair is independent and uniformly random. A *sequence of simulated random generations* is performed similarly, except that *BankInit*, *BankRedraw*, *BankUpdate* are replaced by *SimBankInit*, *SimBankRedraw*, *SimBankUpdate*, and each pair of calls to *BankGen* is replaced by a pair of calls to *SimBankGen* with some specified plaintext bit b (we will always use the same plaintext bit b in both generations).

We now describe several security properties for sequences of real and simulated generations and random generations of pairs. Intuitive descriptions are listed below, and the formal lemma statements follow.

Real and simulated sequences, identical underlying plaintexts. Consider an OC leakage attacker’s “real” view, given leakage from a real sequence of generations using a bank initialized with bit b . Consider also a “simulated” view for the same attacker, given leakage from a simulated sequence of calls, where all calls to *SimBankGen* specify the same underlying plaintext bit b . I.e., the plaintexts underlying the ciphertexts generated in these real and simulated views are all identical. We show that the distributions of the leakage obtained in these two views, *in conjunction with the explicit list of key-ciphertext pairs produced*, are statistically close.

This is stated formally in Lemma 5.1 below. The proof is in Section 5.5.

Two simulated sequences, different underlying plaintexts. Consider an OC leakage attacker’s view, given *two simulated* sequences of generations. The two sequences each produce the same number of ciphertexts, but *differ in the underlying plaintext bits that are specified*.

We show that the distributions of leakage obtained in these two views are statistically close. Note that, unlike the previous property, here statistical closeness does not hold in conjunction with the explicit keys and ciphertexts produced (since the underlying plaintexts differ). We note also that, combining this with the previous property, we conclude statistical closeness of leakage distributions produced by an OC attack on a real sequence and on a simulated sequence with *different underlying plaintexts* (leakage only - without the explicit plaintext and ciphertext produced).

This is stated formally in Lemma 5.2 below. The proof is in Section 5.5.

Single simulated sequence, independence up to orthogonality. Consider an OC leakage attack on a (single) sequence of *simulated* generations. We show that, given the leakage in the attack, the (joint) distribution of keys and ciphertexts produced, is *independent up to orthogonality* (see Definition 3.10). Moreover, the underlying distributions on keys and ciphertexts depend only on the leakage (and the adversary), but not on the sequence of bits given as input to the simulated generations. Finally, these underlying (conditional) distributions have high entropy on each key and each ciphertext produced.

Intuitively, this means that the keys and ciphertexts produced will be resilient to subsequent multi-source leakage. I.e., bounded leakage that operates separately on keys and on ciphertexts will not be able to distinguish the underlying plaintexts. We note that independence up to orthogonality holds even given the list of ciphertexts in the bank in all generations and all randomness used by the ciphertext except the randomness for generating the “target” key and ciphertext.

This is stated formally in Lemma 5.3 below. The proof is in Section 5.5.

Real and simulated sequences of random generations. Consider an OC leakage attacker’s “real” view, given leakage from a real sequence of random generations of ciphertext pairs. Consider also a “simulated” view for the same attacker, given leakage from a simulated sequence of calls, where each pair of calls to *SimBankGen* specify a uniformly random bit (independent of all other pairs). In particular, the plaintexts underlying the ciphertexts generated in these real and simulated views are identically distributed (uniformly random for each pair independently). We show that the distributions of the leakage obtained in these two views, *in conjunction with the explicit list of keys and ciphertext pairs produced*, are statistically close.

This is stated formally in Lemma 5.4 below. This is similar to the guarantee of Lemma 5.1 and we omit the proof.

Single simulated sequence of random generations, independence up to orthogonality.

Consider an OC leakage attack on a (single) sequence of *simulated* random generations of pairs of ciphertexts. We show that, given the leakage in the attack, the (joint) distribution of keys and ciphertexts produced, is *independent up to orthogonality* (see Definition 3.10). Moreover, the underlying distributions on keys and ciphertexts depend only on the leakage (and the adversary), but not on the sequence of underlying plaintext bits. Finally, these underlying (conditional) distributions have high entropy on each key and each ciphertext produced.

Intuitively, this means that the keys and ciphertexts produced will be resilient to subsequent multi-source leakage. I.e., bounded leakage that operates separately on keys and on ciphertexts will not be able to distinguish the underlying plaintexts. Moreover, within each pair of ciphertexts, independence up to orthogonality for the key and each ciphertext separately continues to hold even if the other ciphertext in the pair is released in its entirety. We note that, as was the case above, independence up to orthogonality holds even given the list of ciphertexts in the bank in all generations and all randomness used by the ciphertext except the randomness for generating the “target” key and ciphertext.

This is stated formally in Lemma 5.5 below. The guarantee is quite similar to that of Lemma 5.3 and we omit the proof.

Lemma 5.1. *There exists a leakage bound $\lambda(\kappa) = \Omega(\kappa)$, and a distance bound $\delta(\kappa) = \exp(-\Omega(\kappa))$, s.t. for any bit $b \in \{0, 1\}$, security parameter $\kappa \in \mathbb{N}$, execution bound $T = \text{poly}(\kappa)$, and (computationally unbounded) leakage adversary \mathcal{A} :*

Let Real and Simulated be as follows, where in Real we begin by running $\text{Bank}_0 \leftarrow \text{BankInit}(b)$,

and in *Simulated* we begin by running $Bank_0 \leftarrow SimBankInit$ (both without leakage):

$$\begin{aligned}
Real = \mathcal{A}\{ & ((key_0, \vec{c}_0) \leftarrow BankGen(Bank_0))^{\lambda(\kappa)}, \\
& (Bank_1 \leftarrow BankUpdate(Bank_0))^{\lambda(\kappa)}, key_0, \vec{c}_0, \\
& ((key_1, \vec{c}_1) \leftarrow BankGen(Bank_1))^{\lambda(\kappa)}, \\
& (Bank_2 \leftarrow BankUpdate(Bank_1))^{\lambda(\kappa)}, key_1, \vec{c}_1, \\
& \dots \\
& ((key_{T-1}, \vec{c}_{T-1}) \leftarrow BankGen(Bank_{T-1}))^{\lambda(\kappa)}, \\
& (Bank_T \leftarrow BankUpdate(Bank_{T-1}))^{\lambda(\kappa)}, key_{T-1}, \vec{c}_{T-1} \}
\end{aligned}$$

$$\begin{aligned}
Simulated = \mathcal{A}\{ & ((key_0, \vec{c}_0) \leftarrow SimBankGen(Bank_0, b))^{\lambda(\kappa)}, \\
& (Bank_1 \leftarrow SimBankUpdate(Bank_0))^{\lambda(\kappa)}, key_0, \vec{c}_0, \\
& ((key_1, \vec{c}_1) \leftarrow SimBankGen(Bank_1, b))^{\lambda(\kappa)}, \\
& (Bank_2 \leftarrow SimBankUpdate(Bank_1))^{\lambda(\kappa)}, key_1, \vec{c}_1, \\
& \dots \\
& ((key_{T-1}, \vec{c}_{T-1}) \leftarrow SimBankGen(Bank_{T-1}, b))^{\lambda(\kappa)}, \\
& (Bank_T \leftarrow SimBankUpdate(Bank_{T-1}))^{\lambda(\kappa)}, key_{T-1}, \vec{c}_{T-1} \}
\end{aligned}$$

then $\Delta(Real, Simulated) = \delta(\kappa)$.

Lemma 5.2. *There exists a leakage bound $\lambda(\kappa) = \Omega(\kappa)$, and a distance bound $\delta(\kappa) = \exp(-\Omega(\kappa))$, s.t. for any security parameter $\kappa \in \mathbb{N}$, any execution bound $T = \text{poly}(\kappa)$, any vectors $\vec{b}', \vec{b}'' \in \{0, 1\}^T$, and any (computationally unbounded) leakage adversary \mathcal{A} :*

Let $Simulated'$ and $Simulated''$ be the following two distributions, where in both distributions we

begin by running $Bank_0 \leftarrow SimBankInit$ (without leakage):

$$\begin{aligned}
Simulated' &= \mathcal{A}^{\lambda(\kappa)} \{ [(key_0, \vec{c}_0) \leftarrow SimBankGen(Bank_0, \vec{b}'[0])], \\
&\quad [Bank_1 \leftarrow SimBankUpdate(Bank_0)], \\
&\quad [(key_1, \vec{c}_1) \leftarrow SimBankGen(Bank_1, \vec{b}'[1])], \\
&\quad [Bank_2 \leftarrow SimBankUpdate(Bank_1)], \\
&\quad \dots \\
&\quad [(key_{T-1}, \vec{c}_{T-1}) \leftarrow SimBankGen(Bank_{T-1}, \vec{b}'[T-1])], \\
&\quad [Bank_T \leftarrow SimBankUpdate(Bank_{T-1})] \} \\
Simulated'' &= \mathcal{A}^{\lambda(\kappa)} \{ [(key_0, \vec{c}_0, Bank_1) \leftarrow SimBankGen(Bank_0, \vec{b}''[0])], \\
&\quad [Bank_1 \leftarrow SimBankUpdate(Bank_0)], \\
&\quad [(key_1, \vec{c}_1, Bank_2) \leftarrow SimBankGen(Bank_1, \vec{b}''[1])], \\
&\quad [Bank_2 \leftarrow SimBankUpdate(Bank_1)], \\
&\quad \dots \\
&\quad [(key_{T-1}, \vec{c}_{T-1}, Bank_T) \leftarrow SimBankGen(Bank_{T-1}, \vec{b}''[T-1])], \\
&\quad [Bank_T \leftarrow SimBankUpdate(Bank_{T-1})] \}
\end{aligned}$$

then $\Delta(Simulated', Simulated'') = \delta(\kappa)$.

Lemma 5.3. *There exists a leakage bound $\lambda(\kappa) = \Omega(\kappa)$, and a probability bound $\delta(\kappa) = \exp(-\Omega(\kappa))$, s.t. for any $\kappa \in \mathbb{N}$, any execution bound $T = \text{poly}(\kappa)$, any vector $\vec{b} \in \{0, 1\}^T$, and any (computationally unbounded) leakage adversary \mathcal{A} , the following holds:*

Let $Simulated$ be the following distribution, where we begin by running $Bank_0 \leftarrow SimBankInit$ (without leakage):

$$\begin{aligned}
Simulated &= \mathcal{A}^{\lambda(\kappa)} \{ [(key_0, \vec{c}_0) \leftarrow SimBankGen(Bank_0, \vec{b}[0])], \\
&\quad [Bank_1 \leftarrow SimBankUpdate(Bank_0)], \\
&\quad [(key_1, \vec{c}_1) \leftarrow SimBankGen(Bank_1, \vec{b}[1])], \\
&\quad [Bank_2 \leftarrow SimBankUpdate(Bank_1)], \\
&\quad \dots, \\
&\quad [(key_{T-1}, \vec{c}_{T-1}) \leftarrow SimBankGen(Bank_{T-1}, \vec{b}[T-1])], \\
&\quad [Bank_T \leftarrow SimBankUpdate(Bank_{T-1})] \}
\end{aligned}$$

For any w in the support of $Simulated$, and for any $i \in [T]$, fixing all ciphertexts except the i -th pair produced, let $D_i(w)$ be the joint distribution of (key_i, \vec{c}_i) given w and the remaining $T-1$ ciphertexts. There exist distributions $\mathcal{K}_i(w)$ and $\mathcal{C}_i(w)$ s.t. the following holds:⁷

The distribution $\mathcal{D}_i(w)$ is IuO with orthogonality $\vec{b}[i]$ and underlying distributions $\mathcal{K}_i(w)$ and $\mathcal{C}_i(w)$. With all but $\delta(\kappa)$ probability over the choice (by $Simulated$) of w and of all ciphertexts except the i -th, the min-entropy of $\mathcal{K}_i(w)$ and of $\mathcal{C}_i(w)$ is at least $\kappa - O(\lambda(\kappa))$.

⁷Note that these distributions do not depend on \vec{b}_i (they depend only on w , on \mathcal{A} and on the $T-1$ remaining ciphertexts).

Lemma 5.4. *There exists a leakage bound $\lambda(\kappa) = \Omega(\kappa)$, and a distance bound $\delta(\kappa) = \exp(-\Omega(\kappa))$, s.t. for any security parameter $\kappa \in \mathbb{N}$, execution bound $T = \text{poly}(\kappa)$, and (computationally unbounded) leakage adversary \mathcal{A} :*

Let Real and Simulated be as follows. Choose $\vec{b} \in_R \{0, 1\}^T$. In Real, we begin by running $\text{Bank}_0 \leftarrow \text{BankInit}(\vec{b}[0])$. In Simulated we begin by running $\text{Bank}_0 \leftarrow \text{SimBankInit}$:

$$\begin{aligned} \text{Real} = \mathcal{A}\{ & ((key_0, \vec{c}_0^\alpha) \leftarrow \text{BankGen}(\text{Bank}_0))^{\lambda(\kappa)}, (key_0, \vec{c}_0^\beta \leftarrow \text{BankGen}(\text{Bank}_0))^{\lambda(\kappa)}, \\ & (\text{Bank}'_0 \leftarrow \text{BankRedraw}(\text{Bank}_0))^{\lambda(\kappa)}, (\text{Bank}_1 \leftarrow \text{BankUpdate}(\text{Bank}'_0))^{\lambda(\kappa)}, \\ & key_0, \vec{c}_0^\alpha, \vec{c}_0^\beta, \\ & ((key_1, \vec{c}_1^\alpha) \leftarrow \text{BankGen}(\text{Bank}_1))^{\lambda(\kappa)}, ((key_1, \vec{c}_1^\beta) \leftarrow \text{BankGen}(\text{Bank}_1))^{\lambda(\kappa)} \\ & (\text{Bank}'_1 \leftarrow \text{BankRedraw}(\text{Bank}_1))^{\lambda(\kappa)}, (\text{Bank}_2 \leftarrow \text{BankUpdate}(\text{Bank}'_1))^{\lambda(\kappa)}, \\ & key_1, \vec{c}_1^\alpha, \vec{c}_1^\beta, \\ & \dots \\ & ((key_{T-1}, \vec{c}_{T-1}^\alpha) \leftarrow \text{BankGen}(\text{Bank}_{T-1}))^{\lambda(\kappa)}, ((key_{T-1}, \vec{c}_{T-1}^\beta) \leftarrow \text{BankGen}(\text{Bank}_{T-1}))^{\lambda(\kappa)}, \\ & (\text{Bank}'_{T-1} \leftarrow \text{BankRedraw}(\text{Bank}_{T-1}))^{\lambda(\kappa)}, (\text{Bank}_T \leftarrow \text{BankUpdate}(\text{Bank}'_{T-1}))^{\lambda(\kappa)}, \\ & key_{T-1}, \vec{c}_{T-1}^\alpha, \vec{c}_{T-1}^\beta \} \end{aligned}$$

$$\begin{aligned} \text{Simulated} = \mathcal{A}\{ & ((key_0, \vec{c}_0^\alpha) \leftarrow \text{SimBankGen}(\text{Bank}_0, \vec{b}[0]))^{\lambda(\kappa)}, (key_0, \vec{c}_0^\beta \leftarrow \text{SimBankGen}(\text{Bank}_0, \vec{b}[0]))^{\lambda(\kappa)}, \\ & (\text{Bank}'_0 \leftarrow \text{SimBankRedraw}(\text{Bank}_0))^{\lambda(\kappa)}, (\text{Bank}_1 \leftarrow \text{SimBankUpdate}(\text{Bank}'_0))^{\lambda(\kappa)}, \\ & key_0, \vec{c}_0^\alpha, \vec{c}_0^\beta, \\ & ((key_1, \vec{c}_1^\alpha) \leftarrow \text{SimBankGen}(\text{Bank}_1, \vec{b}[1]))^{\lambda(\kappa)}, ((key_1, \vec{c}_1^\beta) \leftarrow \text{SimBankGen}(\text{Bank}_1, \vec{b}[1]))^{\lambda(\kappa)} \\ & (\text{Bank}'_1 \leftarrow \text{SimBankRedraw}(\text{Bank}_1))^{\lambda(\kappa)}, (\text{Bank}_2 \leftarrow \text{SimBankUpdate}(\text{Bank}'_1))^{\lambda(\kappa)}, \\ & key_1, \vec{c}_1^\alpha, \vec{c}_1^\beta, \\ & \dots \\ & ((key_{T-1}, \vec{c}_{T-1}^\alpha) \leftarrow \text{SimBankGen}(\text{Bank}_{T-1}, \vec{b}[T-1]))^{\lambda(\kappa)}, \\ & ((key_{T-1}, \vec{c}_{T-1}^\beta) \leftarrow \text{SimBankGen}(\text{Bank}_{T-1}, \vec{b}[T-1]))^{\lambda(\kappa)}, \\ & (\text{Bank}'_{T-1} \leftarrow \text{SimBankRedraw}(\text{Bank}_{T-1}))^{\lambda(\kappa)}, (\text{Bank}_T \leftarrow \text{SimBankUpdate}(\text{Bank}'_{T-1}))^{\lambda(\kappa)}, \\ & key_{T-1}, \vec{c}_{T-1}^\alpha, \vec{c}_{T-1}^\beta \} \end{aligned}$$

then $\Delta(\text{Real}, \text{Simulated}) = \delta(\kappa)$.

Lemma 5.5. *There exists a leakage bound $\lambda(\kappa) = \Omega(\kappa)$, and a probability bound $\delta(\kappa) = \exp(-\Omega(\kappa))$, s.t. for any $\kappa \in \mathbb{N}$, any execution bound $T = \text{poly}(\kappa)$, any vector $\vec{b} \in \{0, 1\}^T$, and any (computationally unbounded) leakage adversary \mathcal{A} , the following holds:*

Let *Simulated* be the following distribution, where we begin by running $Bank_0 \leftarrow SimBankInit$:

$$\begin{aligned}
Simulated = \mathcal{A}\{ & ((key_0, \vec{c}_0^\alpha) \leftarrow SimBankGen(Bank_0, \vec{b}[0]))^{\lambda(\kappa)}, (key_0, \vec{c}_0^\beta \leftarrow SimBankGen(Bank_0, \vec{b}[0]))^{\lambda(\kappa)}, \\
& (Bank'_0 \leftarrow SimBankRedraw(Bank_0))^{\lambda(\kappa)}, (Bank_1 \leftarrow SimBankUpdate(Bank'_0))^{\lambda(\kappa)}, \\
& key_0, \vec{c}_0^\alpha, \vec{c}_0^\beta, \\
& ((key_1, \vec{c}_1^\alpha) \leftarrow SimBankGen(Bank_1, \vec{b}[1]))^{\lambda(\kappa)}, ((key_1, \vec{c}_1^\beta) \leftarrow SimBankGen(Bank_1, \vec{b}[1]))^{\lambda(\kappa)}, \\
& (Bank'_1 \leftarrow SimBankRedraw(Bank_1))^{\lambda(\kappa)}, (Bank_2 \leftarrow SimBankUpdate(Bank'_1))^{\lambda(\kappa)}, \\
& key_1, \vec{c}_1^\alpha, \vec{c}_1^\beta, \\
& \dots \\
& ((key_{T-1}, \vec{c}_{T-1}^\alpha) \leftarrow SimBankGen(Bank_{T-1}, \vec{b}[T-1]))^{\lambda(\kappa)}, \\
& ((key_{T-1}, \vec{c}_{T-1}^\beta) \leftarrow SimBankGen(Bank_{T-1}, \vec{b}[T-1]))^{\lambda(\kappa)}, \\
& (Bank'_{T-1} \leftarrow SimBankRedraw(Bank_{T-1}))^{\lambda(\kappa)}, (Bank_T \leftarrow SimBankUpdate(Bank'_{T-1}))^{\lambda(\kappa)}, \\
& key_{T-1}, \vec{c}_{T-1}^\alpha, \vec{c}_{T-1}^\beta \}
\end{aligned}$$

For any w in the support of *Simulated*, and for any $i \in [T]$, fixing all ciphertexts except the i -th pair, let $D_i^\alpha(w)$ and $D_i^\beta(w)$ be the joint distribution of $(key_i, \vec{c}_i^\alpha)$ and (key_i, \vec{c}_i^β) (respectively) given: w , the remaining $T-1$ keys and ciphertext pairs, and (respectively) \vec{c}_i^β and $\langle key_i, \vec{c}_i^\beta \rangle$, or \vec{c}_i^α and $\langle key_i, \vec{c}_i^\alpha \rangle$. Then there exist distributions $\mathcal{K}_i^\alpha(w)$, $\mathcal{C}_i^\alpha(w)$ and $\mathcal{K}_i^\beta(w)$, $\mathcal{C}_i^\beta(w)$ s.t. the following holds:⁸

The distributions $\mathcal{D}_i^\alpha(w)$ and $\mathcal{D}_i^\beta(w)$ are both *IuO* with orthogonality $\vec{b}[i]$ and underlying distributions $\mathcal{K}_i^\alpha(w)$ and $\mathcal{C}_i^\alpha(w)$ or $\mathcal{K}_i^\beta(w)$ and $\mathcal{C}_i^\beta(w)$ (respectively). With all but $\delta(\kappa)$ probability over the choice (by *Simulated*) of the fixed values, the min-entropies of all these underlying distributions are at least $\kappa - O(\lambda(\kappa))$.

5.2 Piecemeal Matrix Computations

Recall that we treat collections of ciphertexts as matrices, where each column of the matrix is a ciphertext. We refer to the procedures in this section as “piecemeal”, because they access the matrices by dividing them into “pieces” or “sketches”, and loading each piece (or sketch) into memory separately. Each piece/sketch is a collection of linear combinations of the matrix’s columns. We refer to these as pieces (rather than sketches) throughout this section.

We present piecemeal procedures for matrix multiplication, for refreshing the key under which the ciphertexts in a matrix’s columns are encrypted, and for adding a vector to the columns of a matrix (we refer to this as matrix-vector addition). We show that these procedures have several security properties under leakage attacks. In all these procedures, no matrix is ever loaded into memory in its entirety. Rather, the matrices are only accessed in a piecemeal manner.

As an (important) example for why this facilitates security, consider the rank of a matrix on which we are computing. If this matrix is loaded into memory in its entirety, then a leakage adversary can compute its rank. If, however, only “pieces” of the matrix are loaded into memory

⁸Note that these distributions *do not depend on* \vec{b}_i (they depend only on w , on \mathcal{A} , on the $T-1$ remaining key-ciphertext pairs, and on the additional i -th ciphertext (\vec{c}_i^β or \vec{c}_i^α respectively)).

at any once time, then it is no longer clear how a leakage adversary can compute the rank. In fact, we will show that (under the appropriate matrix distribution), as long as the matrix is accessed in a piecemeal fashion, its rank is completely hidden, even from a computationally unbounded leakage adversary. This fact will be used extensively in our security proofs. See the subsequent sections for security properties and proofs.

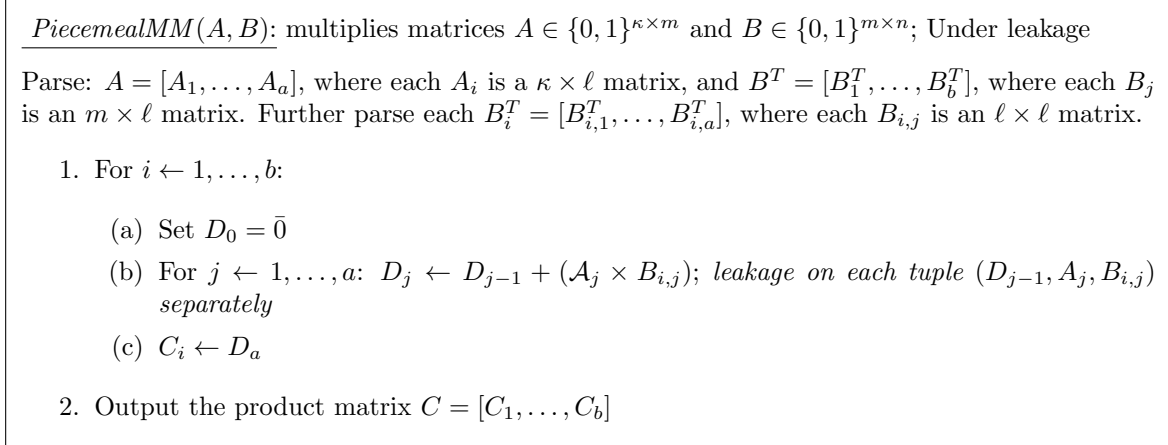


Figure 5: Piecemeal Matrix Multiplication for $\kappa, \ell \in \mathbb{N}$

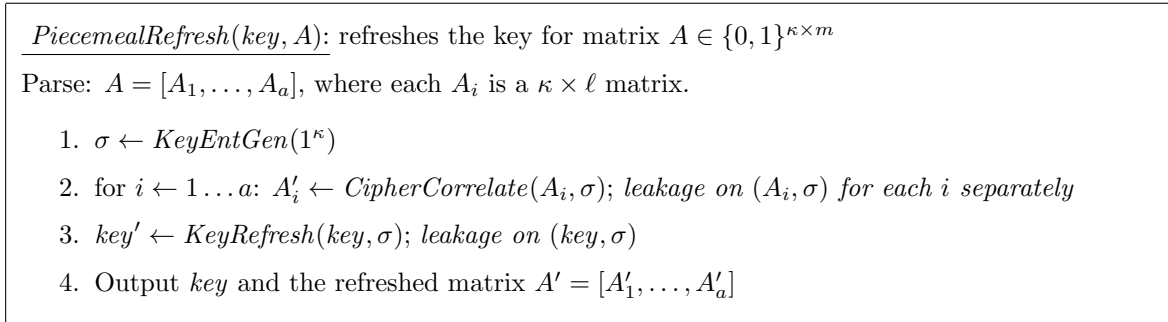


Figure 6: Piecemeal Matrix Refresh for $\kappa, \ell \in \mathbb{N}$

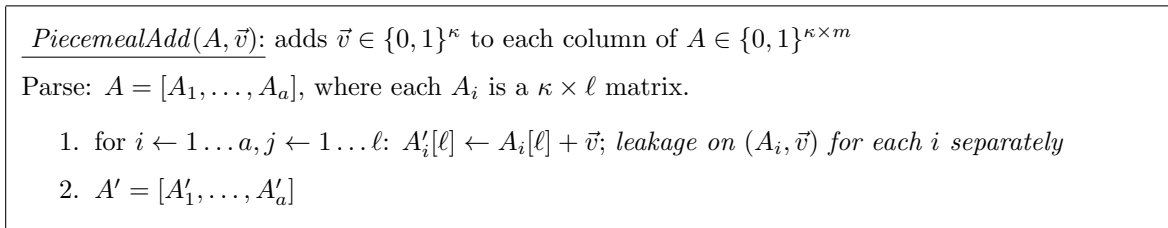


Figure 7: Piecemeal Matrix Addition for $\kappa, \ell \in \mathbb{N}$

5.3 Piecemeal Leakage Attacks on Matrices and Vectors

In this section, we define “piecemeal leakage attacks” on matrices. In particular, these attacks capture the leakage that can be computed via a leakage attack on the piecemeal matrix procedures

(multiplication, refresh, and matrix-vector addition). We prove then that random matrices are resilient to several flavors of such piecemeal attacks.

Attack on a Matrix. A *piecemeal leakage attack* on a matrix, is a multi-source leakage attack, where the sources are *key* and (one or many) “pieces” of the matrix. Recall that each “piece” here is a collection of linear combinations of the matrix columns. See Definition 5.6 below. We focus here on the case where the matrix is either independent of *key*, or has columns orthogonal to *key* (as is the case for a ciphertext bank corresponding to underlying plaintext bit 0). The case where the columns have inner product 1 with *key* is handled similarly.

We will show that a random matrix M is resilient to piecemeal leakage: the leakage computed in such an attack is statistically close when (i) the columns of M are all in the kernel of *key*, (ii) M is a uniformly random matrix, and (iii) M is a uniformly random matrix of rank $\kappa - 1$ (independent of *key*). Moreover, this statistical closeness holds even if *key* is later exposed in its entirety. We begin in Section 5.3.1 with a warmup for the case of an attack on a single piece (Lemma 5.8). We then show security for large number of pieces in Section 5.3.2 (Lemma 5.10).

Definition 5.6 (Piecemeal Leakage Attack on (key, M)). Take $a, \kappa, \lambda, \ell, m \in \mathbb{N}$. Let $\vec{Lin} = (Lin_1, \dots, Lin_a)$ be a sequence of (one or more) matrices, where for each Lin_i , its columns each specify the coefficients of a linear combination of the rows of M . Thus, for $M \in \{0, 1\}^{\kappa \times m}$ and $Lin_i \in \{0, 1\}^{m \times \ell}$, the matrix piece $M \times Lin_i$ is a collection of ℓ linear combinations of M ’s columns.

Let \mathcal{A} be a leakage adversary, operating separately on $key \in \{0, 1\}^\kappa$ and on several matrices in $\{0, 1\}^{\kappa \times \ell}$ (each matrix is $M \times Lin_i$ for some i). We denote \mathcal{A} ’s output by:

$$\mathcal{A}_{\kappa, \ell, m, \vec{Lin}}^\lambda(key, M) \triangleq \mathcal{A}^\lambda(1^\kappa)[key]\{(M \times Lin_1), \dots, (M \times Lin_a)\}$$

we refer to \mathcal{A} as a “piecemeal adversary” operating on (key, M) . We omit κ, λ, ℓ, m and \vec{Lin} when they are clear from the context.

Attack on a Matrix and Vector. We extend these results further, considering piecemeal leakage that operates separately on *key*, and on pieces of a matrix M (as before), each piece jointly with a vector \vec{v} . See Definition 5.7 below.

We show that, for a matrix M with columns in the kernel of *key*, the leakage computed in such an attack is statistically close when (i) the vector \vec{v} is in the kernel of *key*, and (ii) the vector \vec{v} is *not* in the kernel of *key*. Moreover, this statistical closeness holds even if *key* is later exposed in its entirety (as above) *and also* M is later exposed in its entirety. See Section 5.3.3 and Lemma 5.15.

Definition 5.7 (Piecemeal Leakage Attack on $(key, (M, \vec{v}))$). Take $a, \kappa, \lambda, \ell, m \in \mathbb{N}$. Let $\vec{Lin} = (Lin_1, \dots, Lin_a)$ be a sequence of matrices, where for each Lin_i , its columns each specify the coefficients of a linear combination of the rows of M as in Definition 5.6.

Let \mathcal{A} be a leakage adversary, operating separately on $key \in \{0, 1\}^\kappa$ and on several matrices in $\{0, 1\}^{\kappa \times \ell}$ (as in Definition 5.6), each matrix jointly with a vector $\vec{v} \in \{0, 1\}^\kappa$. We denote \mathcal{A} ’s output by:

$$\mathcal{A}_{\kappa, \ell, m, \vec{Lin}}^\lambda(key, (M, \vec{v})) \triangleq \mathcal{A}^\lambda(1^\kappa)[key]\{((M \times Lin_1) \circ \vec{v}), \dots, ((M \times Lin_a) \circ \vec{v})\}$$

we refer to \mathcal{A} as a “piecemeal adversary” operating on $(key, (M, \vec{v}))$. We omit κ, λ, ℓ, m and \vec{Lin} when they are clear from the context.

5.3.1 Piecemeal Leakage Resilience: One Piece

We begin by showing that, for a uniformly random $key \in \{0, 1\}^\kappa$, and a matrix M , given separate leakage from key and from a *single piece* of the matrix, the following two cases induce statistically close distributions. In the first case, the matrix M is uniformly random with columns in the kernel of key . In the second case, M is a uniformly random matrix of rank $\kappa - 1$ (independent of key). By a “single piece” of M we mean any (adversarially chosen) collection of ℓ linear combinations of vectors from M , where here we take $\ell = 0.1\kappa$. This result, stated in Lemma 5.8, is a warm-up for the results in later sections.

Lemma 5.8 (Matrices are Resilient to Piecemeal Leakage with One Piece). *Take $\kappa, m \in \mathbb{N}$ where $m \geq \kappa$. Fix $\ell = 0.1\kappa$ and $\lambda = 0.05\kappa$. Let $Lin \in \{0, 1\}^{m \times \ell}$ be any collection of coefficients for ℓ linear combinations, and \mathcal{A} be any piecemeal leakage adversary. Take $Real$ and $Simulated$ to be the following two distributions:*

$$\begin{aligned} Real &= \left(key, \mathcal{A}_{\kappa, \ell, m, Lin}^\lambda(key, M) \right)_{key \in_R \{0, 1\}^\kappa, M \in_R \{0, 1\}^{\kappa \times m} : \forall i, M[i] \in \text{kernel}(key)} \\ Simulated &= \left(key, \mathcal{A}_{\kappa, \ell, m, Lin}^\lambda(key, M) \right)_{key \in_R \{0, 1\}^\kappa, M \in_R \{0, 1\}^{\kappa \times m} : \text{rank}(M) = \kappa - 1} \end{aligned}$$

then $\Delta(Real, Simulated) \leq 2m \cdot 2^{-0.2\kappa}$.

Remark 5.9. We note that, without any leakage access to key (i.e. given only leakage from the chosen piece of M), a qualitatively similar result to Lemma 5.8 can be derived from a Lemma of Brakerski et al. [BKKV10] on the leakage resilience of random linear subspaces. Their work focused on the more challenging setting where the leakage operates on vectors that are drawn from a low-dimensional subspace (e.g. constant dimension).

Proof of Lemma 5.8. The proof is by a hybrid argument over the matrix columns. For $i \in \{0, \dots, m\}$, let \mathcal{H}_i be the i -th hybrid, where the view is as above but using a matrix M drawn s.t. the first i columns of M_i are uniformly random in the kernel of key , and the last $m - i$ columns are uniformly random s.t. $\text{rank}(M) = \kappa - 1$. We show that for all i , $\Delta(\mathcal{H}_i, \mathcal{H}_{i+1}) \leq 2 \cdot 2^{-0.2\kappa}$. The lemma follows because $\mathcal{H}_0 = Simulated$ and $\mathcal{H}_m = Real$.

We show that the hybrids are close by giving a reduction from the task of predicting the inner product of two vectors under multi-source leakage, to the task of distinguishing \mathcal{H}_i and \mathcal{H}_{i+1} . Since the inner product cannot be predicted under multi-source leakage (by Lemma 3.7), we conclude that the hybrids are statistically close.

To set up the reduction, first fix i . Draw a uniformly random matrix $M \in \{0, 1\}^{\kappa \times m}$ of rank $\kappa - 1$. Let \vec{v} be the $(i+1)$ -th column of M . Let $M_{-(i+1)}$ be the matrix M with the $(i+1)$ -th column set to 0. Now draw $key \in \{0, 1\}^\kappa$ s.t. key is orthogonal to the first i columns in $M_{-(i+1)}$.

We show a reduction from predicting the inner product $\langle key, \vec{v} \rangle$ given multi-source leakage and $(M_{-(i+1)} \times Lin)$, to distinguishing \mathcal{H}_i and \mathcal{H}_{i+1} . This is done by running $\mathcal{A}(key, M)$ on key and on the matrix M drawn above. The reduction computes \mathcal{A} 's (multi-source) leakage on key using multi-source leakage from key . \mathcal{A} 's (multi-source) leakage from $M \times Lin$ is computed using leakage from \vec{v} (since Lin and $M_{-(i+1)} \times Lin$ are “public”). Note now that the joint distribution of (key, M) is exactly as in \mathcal{H}_i . If, however, we condition on the inner product of key and \vec{v} being 0, we get that the joint distribution of (key, M) is exactly as in \mathcal{H}_{i+1} . Thus, if \mathcal{A} has advantage δ in distinguishing \mathcal{H}_i and \mathcal{H}_{i+1} , then the reduction has advantage δ in distinguishing the case that the inner product of key and \vec{v} is 0 from the case that there is no restriction on the inner product.

Now observe that, given $(M_{-(i+1)} \times \text{Lin})$, the vector key is a random variable with min-entropy at least $\kappa - \ell \geq 0.9\kappa$. This is because key is uniformly random under the restriction that it is in the kernel of the first i columns of M . The matrix piece $(M_{-(i+1)} \times \text{Lin})$ contains only $\ell = 0.1\kappa$ vectors, and so it cannot give more than ℓ bits of information on key . Note also that, given $(M_{-(i+1)} \times \text{Lin})$, the $(i+1)$ -th column \vec{v} is independent of key , and also \vec{v} has min entropy at least $(\kappa - 1)$ (in fact \vec{v} has high min entropy even given all of $M_{-(i+1)}$).

The reduction uses $\lambda = 0.05\kappa$ bits of multi-source leakage, and so by lemma 3.8 with all but $2^{-0.2\kappa}$ probability, even given the leakage key and \vec{v} are still independent random sources, both with min entropy at least 0.7κ . When this is the case, by lemma 3.7 we know that, even given key , the inner product of key and \vec{v} is $2^{-0.2\kappa}$ -close to uniform. We conclude that $\delta \leq 2 \cdot 2^{-0.2\kappa}$. The lemma follows. ■

5.3.2 Piecemeal Leakage Resilience: Many Pieces

In this section, we show our main technical result regarding piecemeal matrix leakage. We show that random matrices are resilient to piecemeal leakage on *multiple pieces of the matrix* (operating separately on each piece). In particular, the leakage is statistically close in the case where the matrix is one whose columns are all orthogonal to key and in the case where the matrix is uniformly random. Moreover, this remains true even if key is later exposed in its entirety.

Lemma 5.10 (Matrices are Resilient to Piecemeal Leakage with Many Pieces). *Take $a, \kappa, m \in \mathbb{N}$, where $m \geq \kappa$. Fix $\ell = 0.1\kappa$, and $\lambda = 0.05\kappa/a$. Let $\vec{\text{Lin}} = (\text{Lin}_1, \dots, \text{Lin}_a)$ be any sequence of collections of coefficients for linear combinations, where for each i , $\text{Lin}_i \in \{0, 1\}^{m \times \ell}$ has full rank ℓ . Let \mathcal{A} be any piecemeal leakage adversary. Take *Real* and *Simulated* to be the following two distributions:*

$$\begin{aligned} \text{Real} &= \left(\text{key}, \mathcal{A}_{\kappa, \ell, m, \vec{\text{Lin}}}^{\lambda}(\text{key}, \mathbf{M}) \right)_{\text{key} \in_R \{0, 1\}^{\kappa}, \mathbf{M} \in_R \{0, 1\}^{\kappa \times m} : \forall i, \mathbf{M}[i] \in \text{kernel}(\text{key})} \\ \text{Simulated} &= \left(\text{key}, \mathcal{A}_{\kappa, \ell, m, \vec{\text{Lin}}}^{\lambda}(\text{key}, \mathbf{M}) \right)_{\text{key} \in_R \{0, 1\}^{\kappa}, \mathbf{M} \in_R \{0, 1\}^{\kappa \times m} : \text{rank}(\mathbf{M}) = \kappa - 1} \end{aligned}$$

then $\Delta(\text{Real}, \text{Simulated}) \leq 5a^2 \cdot 2^{-0.04\kappa/a}$.

Proof. For $i \in \{0, \dots, a\}$, we denote $P_i = M \times \text{Lin}_i$ the matrix “piece” being leaked on/attacked in the i -th part of the attack. We use w_i to denote the leakage accumulated by \mathcal{A} up to and including the i -th attack. We will consider \mathcal{V}_i , the conditional distribution on (key, M) , drawn as in *Real*, given the leakage w_i . Namely, in \mathcal{V}_0 we have key drawn uniformly at random and M is random with columns in $\text{kernel}(\text{key})$. Note that the random variables key and M , when drawn by \mathcal{V}_i , are not independent. In particular, key and the columns of M are orthogonal. Let \mathcal{K}_i and \mathcal{M}_i be the marginal distributions of \mathcal{V}_i on key and on M .

Hybrids. We will prove Lemma 5.10 using a hybrid argument. For $i \in \{0, \dots, a\}$, we define a hybrid distribution \mathcal{H}_i . Each hybrid’s output domain will be $\text{key} \in \{0, 1\}^{\kappa}$ and leakage values computed by $\mathcal{A}(\text{key}, M)$.

For each i , we define \mathcal{H}_i by drawing $(\text{key}, M) \sim \mathcal{V}_0$, and simulating the piecemeal leakage attack $\mathcal{A}(\text{key}, M)$. We always use key for computing the key leakage in the attack. For leakage on the j -th matrix piece, however, we use P_j ’s drawn differently for each \mathcal{H}_i :

- For $j \in \{1, \dots, i\}$, we define $P_j = (\mathbf{M} \times \text{Lin}_j)$.
- For $j \in \{i+1, \dots, a\}$, re-draw $M_j \sim \mathcal{M}_{j-1}$. I.e., we re-draw the matrix from the current marginal distribution of \mathcal{V}_{j-1} on M , independently of key . Define $P_j = (M_j \times \text{Lin}_j)$.

Clearly, $\mathcal{H}_a = \text{Real}$, because in \mathcal{H}_a we never compute leakage on a re-drawn matrix M_j . We will show that $\mathcal{H}_0 = \text{Simulated}$, see Claim 5.11. Note that this is non-trivial because in \mathcal{H}_0 the matrix M is continually re-drawn from M_j (independently of key), whereas in Simulated the matrix M is never redrawn. Nonetheless, Claim 5.11 below shows that, because the leakage operates separately on key and on M , these two distributions are identical.

Claim 5.11. $\mathcal{H}_0 = \text{Simulated}$

Proof of Claim 5.11. Fix leakage w_j for the first j attacks on pieces of M . In the distribution \mathcal{H}_0 , for the $(j+1)$ -th matrix piece, we use $P_{j+1} = M_{j+1} \times \text{Lin}_{j+1}$, where M_{j+1} is re-drawn from the marginal distribution \mathcal{M}_j .

In the distribution Simulated , on the other hand, we use $P_{j+1} = M \times \text{Lin}_{j+1}$, where M is drawn from \mathcal{M}'_j , the distribution of uniformly random M 's of rank $\kappa - 1$ (independent of key), given that the multi-source leakage so far was w_j .

Other than this difference, the distributions are identical. Thus, it suffices to show that, for every j and every fixed leakage w_j in the first j attacks, we have that $\mathcal{M}_j = \mathcal{M}'_j$.

The leakage in the first j attacks operates separately on key and on M . Thus, we know that conditioning the joint distribution \mathcal{V}_0 on w_j , is equivalent to conditioning \mathcal{V}_0 on (key, M) falling in a product set. Let $S_{key} \subseteq \{0, 1\}^\kappa$ and $S_M \subseteq \{0, 1\}^{\kappa \times 2\kappa}$ be the sets s.t. for all $(key, M) \in S_{key} \times S_M$, the leakage on the first j pieces in a piecemeal attack on (key, M) equals w_j . Now we know that \mathcal{M}_j is exactly equal to \mathcal{M}_0 , conditioned on M falling in the set S_M .

Similarly, in Simulated the distribution \mathcal{M}'_j is the uniform distribution on rank $\kappa - 1$ matrices, conditioned on the leakage w_j , i.e. on M falling in the set S_M . Since \mathcal{M}_0 is uniform on rank $\kappa - 1$ matrices, for any w_j we get that $\mathcal{M}_j = \mathcal{M}'_j$. The claim follows. ■

To complete the proof of Lemma 5.10, we will show that $\Delta(\mathcal{H}_i, \mathcal{H}_{i+1}) \leq 4m \cdot 2^{-0.04\kappa/a}$. The lemma follows by a hybrid argument. For this, consider the joint distribution of key , and of the leakage w_{i+1} computed on the first $(i+1)$ pieces. We will show that the joint distribution is statistically close in both hybrids. This suffices to show that the hybrids themselves are statistically close, because, for both hybrids, the leakage on pieces $((i+2), \dots, a)$, and the remaining leakage on key , can be computed as a function of (key, w_{i+1}) (the same function for both hybrids).

In both $\mathcal{H}_i, \mathcal{H}_{i+1}$, leakage on the first i pieces is computed in exactly the same way. The difference is in leakage on the $(i+1)$ -th piece. Fixing the leakage w_i on the first i pieces, in \mathcal{H}_{i+1} we have P_{i+1} computed using dependent $(key, M) \sim \mathcal{V}_i$. In \mathcal{H}_i we use independent $key \sim \mathcal{K}_i, M \sim \mathcal{M}_i$. These two different distributions yield different leakage w on the $(i+1)$ -th piece.

Piecemeal Leakage from IuO Distributions. key and M drawn (jointly) by \mathcal{V}_i are not independent. In general, for a dependant distribution \mathcal{V}_i on key and M with marginal distributions \mathcal{K}_i and \mathcal{M}_i , leakage on $(key, M) \sim \mathcal{V}_i$ could look very different from leakage on $(key \sim \mathcal{K}_i, M \sim \mathcal{M}_i)$. We will show, however, that piecemeal leakage resilience *does hold* in a special case where the joint distribution \mathcal{V}_i is independent up to orthogonality (IuO, see Definition 3.10). We will also show it holds when \mathcal{V}_i is statistically close to IuO, as defined below.

Definition 5.12 (Key-Matrix α -Independence up to Orthogonality). Let \mathcal{V} be a distribution on pairs (key, M) , where $key \in \{0, 1\}^\kappa$, $M \in \{0, 1\}^{\kappa \times 2\kappa}$ and M is always of rank $\kappa - 1$. We say that \mathcal{V} is α -independent up to orthogonality, if there exists distribution \mathcal{V}' that is independent up to orthogonality and $\Delta(\mathcal{V}, \mathcal{V}') \leq \alpha$.

We will show that piecemeal leakage on an IuO distribution is statistically close to piecemeal leakage when key and M are sampled from the independently drawn variant, see Claim 5.13 below. We also show that \mathcal{V}_i is (w.h.p over w_i) an IoU distribution, see Claim 5.14. Statistical closeness of the hybrids \mathcal{H}_i and \mathcal{H}_{i+1} follows.

Claim 5.13. Take $a, \kappa, m, \ell, \lambda$ as in Lemma 5.10. Let \mathcal{V} be any distribution over pairs (key, M) , where $key \in \{0, 1\}^\kappa$, $M \in \{0, 1\}^{\kappa \times m}$ and M has rank $\kappa - 1$. Suppose that \mathcal{V} is IuO, with underlying distributions \mathcal{K} and \mathcal{M} . Suppose further that \mathcal{V} has min-entropy at least $(\kappa + (\kappa - 1) \cdot 2\kappa - 0.15\kappa)$.

Let $Lin \in \{0, 1\}^{m \times \ell}$ be a collection of coefficients for linear combinations, specified by a matrix of rank ℓ . Let \mathcal{A} be any piecemeal leakage adversary. Take \mathcal{D} and \mathcal{F} to be the following distributions:

$$\begin{aligned}\mathcal{D} &= (key, w)_{(key, M) \sim \mathcal{V}, w \leftarrow \mathcal{A}(key, M)} \\ \mathcal{F} &= (key, w)_{key \sim \mathcal{K}, M \sim \mathcal{M}, w \leftarrow \mathcal{A}(key, M)}\end{aligned}$$

Take $\delta = (4\ell \cdot 2^{-0.05\kappa})$. Then $\Delta(D, F) \leq 2\delta$. Moreover, with all but δ probability over $w \sim D$, we have that $\Delta((D|\mathcal{A}(key, M) = w), (F|\mathcal{A}(key, M) = w)) \leq \delta$.

The proof of Claim 5.13 is below.

Claim 5.14. Take $a, \kappa, \ell, \lambda, \mathcal{V}, L, \mathcal{A}$ as in Claim 5.13. Suppose here that \mathcal{V} : (i) has min-entropy at least $(\kappa + (\kappa - 1) \cdot 2\kappa - 0.15\kappa)$ (as in Claim 5.13), and (ii) is α -close to independence up to orthogonality (see Definition 5.12). Define the distribution:

$$\mathcal{V}(w) = (key, M)_{(key, M) \sim \mathcal{V}: \mathcal{A}(key, M) = w}$$

and take $\delta = (4\ell \cdot 2^{-0.05\kappa})$. For any $0 < \beta < 1$, with all but $(\beta + \delta)$ probability over $w \leftarrow \mathcal{A}(key, M)_{(key, M) \sim \mathcal{V}}$ it is the case that $\mathcal{V}(w)$ is $((\alpha/\beta) + \delta)$ -close to independence up to orthogonality.

The proof of Claim 5.14 is below. We now complete the proof of Lemma 5.10:

1. With all but $2^{-0.05\kappa}$ probability over w_i , for all $j \leq i$ simultaneously, the min-entropy of \mathcal{V}_j is at least $\kappa + (\kappa - 1) \cdot 2\kappa - 0.15\kappa$. This is by Lemma 3.8, because the min-entropy of \mathcal{V}_0 is $\kappa + (\kappa - 1) \cdot 2\kappa$, and the amount of leakage in the first $i \leq a$ attacks (leakage from both key and M) is less than 0.1κ .
2. Take $\delta = (4\ell \cdot 2^{-0.05\kappa})$, $\beta = 2^{-0.04\kappa/a}$. We show the following by induction for $j \leq i$:
with all but $(2^{-0.05\kappa} + j \cdot (\delta + \beta))$ probability over w_i , we have that \mathcal{V}_j is $(2\delta/\beta^j)$ -close to independence up to orthogonality (and also the min entropy bound of Item 1 holds). The induction basis follows because \mathcal{V}_0 is perfectly independent up to orthogonality. The induction step follows from Claim 5.14 (and the min-entropy bound in Item 1).

Finally, we use Claim 5.13 to conclude that with all but $(2^{-0.05\kappa} + i \cdot (\delta + \beta))$ probability over w_i , the hybrids \mathcal{H}_i and \mathcal{H}_{i+1} are $(2\delta/\beta^i + 2\delta)$ -statistically close. In particular, this implies that

$$\Delta(\mathcal{H}_i, \mathcal{H}_{i+1}) \leq (2^{-0.05\kappa} + i \cdot (\delta + \beta)) + (2\delta/\beta^i) + 2\delta \leq 5a \cdot 2^{-0.04\kappa/a}$$

where the second inequality assumes $i \cdot \beta$ is the largest term in the sum (and using $i \leq a$). \blacksquare

Proof of Claim 5.13. The proof is by a hybrid argument. We denote $P = M \times L$. For $i \in [a + 1]$, take the i -th hybrid \mathcal{H}_i to be:

$$\mathcal{H}_i = (key, w)_{M \sim \mathcal{M}, P \leftarrow M \times L, key \sim (\mathcal{K} | P[1], \dots, P[i]), w \leftarrow \mathcal{A}(key, P)}$$

i.e. the key is drawn from a conditional distribution on \mathcal{K} , conditioning on the first i columns of P . We get that $\mathcal{H}_0 = \mathcal{F}$, because key is drawn without conditioning on any columns (i.e. independently of M). Also $\mathcal{H}_\ell = \mathcal{D}$, because key is re-drawn conditioned on all of P , which is the same as just drawing $(key, M) \sim \mathcal{V}$ and taking $P = M \times L$.

For each pair of hybrids, we bound $\Delta(\mathcal{H}_i, \mathcal{H}_{i+1})$. To do so, consider the following experiment: draw $(P[1], \dots, P[i]) \sim M$ (as in both \mathcal{H}_i and \mathcal{H}_{i+1}). Fixing these draws, in \mathcal{H}_i the distribution of $P[i + 1]$ is a random sample from $\mathcal{P}_i = (P[i + 1] | M \sim \mathcal{M} | P[1], \dots, P[i])$. Similarly, in \mathcal{H}_i we have that key is a random sample from $\mathcal{K}_i = (\mathcal{K} | P[1], \dots, P[i])$. In particular, note that key is independent of $P[i + 1]$.

We now examine \mathcal{H}_i^+ , obtained from \mathcal{H}_i by including also the inner product of key and $P[i + 1]$. We can also consider \mathcal{H}_i^R , obtained from \mathcal{H}_i by adding a uniformly random bit:

$$\begin{aligned} \mathcal{H}_i^+ &= (key, \langle key, P[i + 1] \rangle, w)_{key \sim \mathcal{K}_i, P[i+1] \sim \mathcal{P}_i, (P[i+2], \dots, P[\ell]) \sim (\mathcal{M} | P[1], \dots, P[i+1]), w \leftarrow \mathcal{A}(key, P)} \\ \mathcal{H}_i^R &= (key, \mathbf{r}, w)_{key \sim \mathcal{K}_i, P[i+1] \sim \mathcal{P}_i, (P[i+2], \dots, P[\ell]) \sim (\mathcal{M} | P[1], \dots, P[i+1]), w \leftarrow \mathcal{A}(key, P), \mathbf{r} \in \mathbf{R}\{0, 1\}} \end{aligned}$$

We will show that $\Delta(\mathcal{H}_i, \mathcal{H}_{i+1}) \leq 2\Delta(\mathcal{H}_i^+, \mathcal{H}_i^R)$. To show this, consider now \mathcal{H}_{i+1} . Again, $P[i + 1]$ is an independent sample from \mathcal{P}_i (as in \mathcal{H}_i). Here, however, we have that key depends on $P[i + 1]$ and is a sample from $\mathcal{K}_{i+1} = (\mathcal{K} | w, P[1], \dots, P[i], \mathbf{P}[i + 1])$. Since \mathcal{V} is independent up to orthogonality, we have:

$$\begin{aligned} \mathcal{K}_{i+1} &= (key, P[1], \dots, P[i], P[i + 1])_{(key, M) \sim \mathcal{V}, P \leftarrow M \times L} \\ &= (key, P[1], \dots, P[i], \langle key, \mathbf{P}[i + 1] \rangle = \mathbf{0})_{(key, M) \sim \mathcal{V}, P \leftarrow M \times L} \end{aligned}$$

given $(key, P[1], \dots, P[i + 1])$, the marginal distributions of $(P[i + 2], \dots, P[\ell])$ and of w in \mathcal{H}_{i+1} are identical to \mathcal{H}_i . Thus, the only difference between \mathcal{H}_i and \mathcal{H}_{i+1} is that in \mathcal{H}_{i+1} we add an extra condition on key to be in the kernel of $P[i + 1]$.

Re-examining \mathcal{H}_i^+ , by definition \mathcal{H}_i is the marginal distribution of \mathcal{H}_i^+ on (key, w) . We now conclude also that \mathcal{H}_{i+1} is the marginal distribution on (key, w) in \mathcal{H}_i^+ conditioned on $\langle key, P[i + 1] \rangle = 0$. Thus $\Delta(\mathcal{H}_i, \mathcal{H}_{i+1}) \leq 2\Delta(\mathcal{H}_i^+, \mathcal{H}_i^R)$.

It remains to bound $\Delta(\mathcal{H}_i^+, \mathcal{H}_i^R)$. We know that in both these distributions, given $(P[1], \dots, P[i])$ (without w), we have that key and $P[i + 1]$ are drawn independently and the joint distribution of $(key, P[i + 1])$ has entropy at least $(1.85\kappa - i) \geq 1.75\kappa$. This is simply by the min-entropy of \mathcal{V} . By Lemma 3.8, with all but $2^{-0.05\kappa}$ probability over the choice of w , the min-entropy of $(key, P[i + 1])$ given also w (of length at most 0.1κ) is at least 1.6κ .

We conclude, by Lemma 3.7, that with all but $2^{-0.05\kappa}$ probability over $w \sim \mathcal{H}_i$, it is the case that with all but $2^{-0.05\kappa}$ probability over key conditioned on w , the inner product of key and $P[i + 1]$ (given (key, w)) is $2^{-0.05\kappa}$ -close to uniform. In particular, when this is the case, with all but $2 \cdot 2^{-0.05\kappa}$ probability over $(key, w) \sim \mathcal{H}_i$, we have that the probabilities of (key, w) by \mathcal{H}_i and by \mathcal{H}_{i+1} differ by at most a $\exp(1.5 \cdot 2^{-0.05\kappa})$ multiplicative factor. The claim follows. ■

Proof of Claim 5.14. \mathcal{V} is α -close to IuO. Let \mathcal{V}' be an IuO distribution s.t. $\Delta(\mathcal{V}, \mathcal{V}') \leq \alpha$. Let \mathcal{K}' and \mathcal{M}' be the marginal distributions of \mathcal{V}' on key and M (respectively). Now take:

$$\begin{aligned}\mathcal{Z}' &\triangleq (key, M, w)_{(key, M) \sim \mathcal{V}', w \leftarrow \mathcal{A}(key, M)}, \\ &= (key, \mathbf{M}, w)_{(key, \mathbf{M}') \sim \mathcal{V}', w \leftarrow \mathcal{A}(key, M'), \mathbf{M} \sim (\mathcal{M}' | key, \mathcal{A}(key, M) = w)} \\ \mathcal{Z}'' &\triangleq (key, \mathbf{M}, w)_{key \sim \mathcal{K}', \mathbf{M}' \sim \mathcal{M}', w \leftarrow \mathcal{A}(key, M'), \mathbf{M} \sim (\mathcal{M}' | key, \mathcal{A}(key, M) = w)}\end{aligned}$$

Let $\mathcal{Z}'(w)$ and $\mathcal{Z}''(w)$ be the marginal distributions of \mathcal{Z}' and \mathcal{Z}'' (respectively) on (key, M) , conditioned on $\mathcal{A}(key, M) = w$. Note that $\mathcal{Z}'(w)$ is also the conditional distribution of \mathcal{V}' (conditioned on w). By Claim 5.13, we know that with all but δ probability over $w \sim \mathcal{Z}'$ we have that $\Delta(\mathcal{Z}'(w), \mathcal{Z}''(w)) \leq \delta$. Claim 5.13 shows this is true for the marginal distributions on (key, w) , but in \mathcal{Z}' and \mathcal{Z}'' , the matrix M is just a probabilistic function of (key, w) , and so the bound on the statistical distance holds also when M is added to the output.

We claim that (for any w), the distribution $\mathcal{Z}''(w)$ is (perfectly) independent up to orthogonality. This is because in \mathcal{Z}'' , the leakage w is computed as multi-source leakage on independently drawn key and M . Thus, conditioning \mathcal{Z}'' on w is conditioning \mathcal{Z}'' on (key, M) falling in a product set $S_{key} \times S_M$. We know that \mathcal{Z}'' is (perfectly) independent up to orthogonality, and so conditioning \mathcal{Z}'' on a product set $S_{key} \times S_M$ will also yield a distribution that is independent up to orthogonality.

We conclude that, with all but δ probability over $w \sim \mathcal{Z}'$, we have that $\Delta(\mathcal{Z}'(w), \mathcal{Z}''(w)) \leq \delta$ and $\mathcal{Z}''(w)$ is independent up to orthogonality. Let W_{bad} be the set of “bad” w ’s for which $\Delta(\mathcal{Z}'(w), \mathcal{Z}''(w)) > \delta$. Since $\Delta(\mathcal{V}, \mathcal{V}') \leq \alpha$, we know that:

$$\begin{aligned}\Pr_{w \sim \mathcal{V}} [w \in W_{bad}] &\leq \alpha + \delta \\ \Pr_{w \sim \mathcal{V}} [\Delta(\mathcal{V}(w), \mathcal{V}'(w)) \geq (\alpha/\beta)] &\leq \beta\end{aligned}$$

where the second equation follows by Markov’s inequality. We conclude (by a union bound, and since $\mathcal{V}'(w) = \mathcal{Z}'(w)$), that with all but $(\alpha + \beta + \delta)$ probability over $w \sim \mathcal{V}$, we have that $\mathcal{V}(w)$ is $((\alpha/\beta) + \delta)$ -close to $\mathcal{Z}''(w)$ and to independence up to orthogonality. ■

5.3.3 Piecemeal Leakage Resilience: Jointly with a Vector

In this section, we show further security properties of random matrices under piecemeal leakage. We focus on piecemeal leakage that operates jointly on (each piece of) a matrix and a vector (and separately on key). The matrix will always have columns that are (random) in the kernel of key . We show that the leakage is statistically close in the cases where the vector is and is not in the kernel. Moreover, this statistical closeness is *strong* and holds even if the matrix is later released *in its entirety*. The proof is based on Lemma 5.10 (piecemeal leakage resilience of random matrices) and on a “pairwise independence” property under piecemeal leakage, stated separately in Claim 5.16 below.

Lemma 5.15 (Strong Resilience to Matrix-Vector Piecemeal Leakage). *Take $a, \kappa, m \in \mathbb{N}$, where $m \geq \kappa$. Fix $\ell = 0.1\kappa$, and $\lambda = 0.01\kappa/a^2$. Let $\vec{Lin} = (Lin_1, \dots, Lin_a)$ be any sequence of collections of coefficients for linear combinations, where for each i , $Lin_i \in \{0, 1\}^{m \times \ell}$ has full rank ℓ . Let \mathcal{A} be any piecemeal leakage adversary. Take Real and Simulated to be the following two distributions:*

$$\begin{aligned}\text{Real} &= \left(key, M, \mathcal{A}_{\kappa, \ell, m, \vec{Lin}}^\lambda(key, (M, \vec{v})) \right)_{key \in_R \{0, 1\}^\kappa, M \in_R \{0, 1\}^{\kappa \times m} : \forall i, M[i] \in \text{kernel}(key), \vec{v} \in_R \text{kernel}(key)} \\ \text{Simulated} &= \left(key, M, \mathcal{A}_{\kappa, \ell, m, \vec{Lin}}^\lambda(key, (M, \vec{v})) \right)_{key \in_R \{0, 1\}^\kappa, M \in_R \{0, 1\}^{\kappa \times m} : \forall i, M[i] \in \text{kernel}(key), \vec{v} \in_R \overline{\text{kernel}(key)}}\end{aligned}$$

then $\Delta(\text{Real}, \text{Simulated}) \leq 3a \cdot 2^{-0.01\kappa/a}$.

Proof. We define the “midpoint” distribution:

$$\mathcal{D} = 1/2 \cdot \text{Real} + 1/2 \cdot \text{Simulated} = (\text{key}, M, w = \mathcal{A}(\text{key}, (M, \vec{v})))_{\text{key}, M, \vec{v} \in_R \{0,1\}^\kappa}$$

For fixed (key, M, w) , we consider their *bias*:

$$\text{bias}(\text{key}, M, w) \triangleq \frac{\text{Real}[\text{key}, M, w] - \text{Simulated}[\text{key}, M, w]}{\mathcal{D}[\text{key}, M, w]}$$

And note that (by definition):

$$\Delta(\text{Real}, \text{Simulated}) = \mathbb{E}_{(\text{key}, M, w) \sim \mathcal{D}}[|\text{bias}(\text{key}, M, w)|]/2 \quad (1)$$

Thus we focus on bounding $\mathbb{E}_{(M, w) \sim H}[|\text{bias}(\text{key}, M, w)|]$. We will use a “pairwise independence” property of matrices under piecemeal leakage.

Claim 5.16 (Pairwise Independence under Piecemeal Leakage). *Take $a, \kappa, m, \ell, \lambda, \vec{Lin}, \mathcal{A}$ as in Lemma 5.16. Let \mathcal{F} and \mathcal{F}' be the following distributions. In both \mathcal{F} and \mathcal{F}' , take $\text{key} \in_R \{0,1\}^\kappa$, a matrix $M \in_R \{0,1\}^{\kappa \times m}$ s.t. all of M ’s columns are in the kernel of key . Choose $\vec{v}_1, \vec{v}_2 \in_R \{0,1\}^\kappa$ s.t. $\mathcal{A}(\text{key}, (M, \vec{v}_1)) = \mathcal{A}(\text{key}, (M, \vec{v}_2))$.*

$$\begin{aligned} \mathcal{F} &= (\vec{v}_1, \vec{v}_2, b_1, b_2, \mathcal{A}(\text{key}, (M, \vec{v}_1)))_{\text{key}, M, \vec{v}_1, \vec{v}_2, \mathbf{b}_1 = \langle \text{key}, \vec{v}_1 \rangle, \mathbf{b}_2 = \langle \text{key}, \vec{v}_2 \rangle} \\ \mathcal{F}' &= (\vec{v}_1, \vec{v}_2, b_1, b_2, \mathcal{A}(\text{key}, (M, \vec{v}_1)))_{\text{key}, M, \vec{v}_1, \vec{v}_2, \mathbf{b}_1, \mathbf{b}_2 \in_R \{0,1\}} \end{aligned}$$

then $\Delta(\mathcal{F}, \mathcal{F}') \leq \delta = 5a^2 \cdot 2^{-0.03\kappa/a}$.

The proof of Claim 5.16 is below.

We will show that if $\mathbb{E}_{(M, w) \sim H}[|\text{bias}(\text{key}, M, w)|]$ is too high, then we can predict the inner products of \vec{v}_1, \vec{v}_2 as above with key and distinguish \mathcal{F} and \mathcal{F}' (a contradiction to Claim 5.16). We do this by considering a distinguisher \mathcal{DLS} that gets $(\vec{v}_1, \vec{v}_2, b_1, b_2, w)$ (where $(\vec{v}_1, \vec{v}_2, w)$ are distributed as in both \mathcal{F} and \mathcal{F}'), and attempts to distinguish whether $b_1, b_2 \in \{0,1\}$ are uniformly random (distribution \mathcal{F}'), or are the inner products of \vec{v}_1, \vec{v}_2 with key (distribution \mathcal{F}). The distinguisher \mathcal{DLS} outputs 1 if $b_1 = b_2$ and outputs 0 otherwise. By Claim 5.16, the advantage of (any distinguisher, and in particular also of) \mathcal{DLS} is bounded by $\delta = 6a^2 \cdot 2^{-0.03\kappa}$.

For distribution \mathcal{F}' , the bits b_1, b_2 are independent uniform bits, and so the probability that \mathcal{DLS} outputs 1 is exactly $1/2$. In distribution \mathcal{F} , however, if $\mathbb{E}_{(M, w) \sim \mathcal{D}}[|\text{bias}(\text{key}, M, w)|]$ is high then \mathcal{DLS} will output 1 with significantly higher probability (this gives a bound on the expected magnitude of the bias).

To see this, fix (key, M) . For a possible leakage value $w \in \{0,1\}^{a\lambda}$, denote by $p_{\text{key}, M, w}$ the probability of leakage w given key and M (for $(\text{key}, M, \vec{v}) \sim \mathcal{D}$). Conditioning \mathcal{D} on (key, M) , the probability of identical leakage from uniformly random \vec{v}_1 and \vec{v}_2 is the “collision probability” $cp(\text{key}, M) \triangleq \sum_{w \in \{0,1\}^{a\lambda}} p_{\text{key}, M, w}^2$. Conditioning \mathcal{D} on (key, M) and identical leakage from \vec{v}_1 and \vec{v}_2 , the probability that the leakage is some specific value w is exactly $p_{\text{key}, M, w}^2 / cp(\text{key}, M)$. Conditioning \mathcal{D} on (key, M) and identical leakage w from \vec{v}_1, \vec{v}_2 , the probability that the inner products of \vec{v}_1 and \vec{v}_2 with key are equal and \mathcal{DLS} outputs 1 is exactly $1/2 + 2|\text{bias}(\text{key}, M, w)|^2$

(notice that the advantage over $1/2$ is always “in the same direction”). Since (by Claim 5.16) the advantage of \mathcal{DIS} is at most δ , we get that:

$$\begin{aligned}\delta &\geq E_{key,M} [\mathcal{DIS}'\text{'s advantage in outputting 1 given } (key, M)] \\ &= E_{key,M} \left[\sum_{w \in \{0,1\}^{a \cdot \lambda}} (p_{key,M,w}^2 / cp(key, M)) \cdot 2|bias(key, M, w)|^2 \right]\end{aligned}$$

Now because $cp(key, M) \geq 2^{-a \cdot \lambda}$, we get that:

$$E_{key,M} \left[\sum_{w \in \{0,1\}^{a \cdot \lambda}} p_{key,M,w}^2 \cdot 2|bias(key, M, w)|^2 \right] \leq 2^{a \cdot \lambda} \cdot \delta \quad (2)$$

We also have that:

$$\begin{aligned}2\Delta(Real, Simulated) &= E_{(key,M,w) \sim \mathcal{H}} [|bias(key, M, w)|] \\ &= E_{key,M} \left[\sum_{w \in \{0,1\}^{a \cdot \lambda}} p_{key,M,w} \cdot |bias(key, M, w)| \right] \\ &\leq \sqrt{2^{a \cdot \lambda} \cdot E_{key,M} \left[\sum_{w \in \{0,1\}^{a \cdot \lambda}} p_{key,M,w}^2 \cdot |bias(key, M, w)|^2 \right]}\end{aligned}$$

where the last inequality is by Cauchy-Schwartz. Putting this together with Equation 2, we get:

$$\Delta(Real, Simulated) \leq 2^{a \cdot \lambda} \cdot \sqrt{\delta} < 3a \cdot 2^{-0.01\kappa/a}$$

which completes the proof. \blacksquare

Proof of Claim 5.16. Consider the following distribution \mathcal{E} , where key is uniformly random, M is a uniformly random matrix with columns in key 's kernel, and \vec{v}_1, \vec{v}_2 are uniformly random pair s.t. $\mathcal{A}(key, (M, \vec{v}_1)) = \mathcal{A}(key, (M, \vec{v}_2))$:

$$\mathcal{E} = (key, \vec{v}_1, \vec{v}_2, \mathcal{A}(key, (\mathbf{M}, \vec{v}_1)))_{key, \mathbf{M} \in_R \{0,1\}^{\kappa \times m} : \forall i, M[i] \in \text{kernel}(key), \vec{v}_1, \vec{v}_2}$$

Consider also the distribution \mathcal{H} that uses a uniformly random matrix M of rank $\kappa - 1$:

$$\mathcal{H} = (key, \vec{v}_1, \vec{v}_2, \mathcal{A}(key, (\mathbf{M}, \vec{v}_1)))_{key, \mathbf{M} \in_R \{0,1\}^{\kappa \times m} : \text{rank}(M) = \kappa - 1, \vec{v}_1, \vec{v}_2}$$

We will show that:

1. $\Delta(\mathcal{E}, \mathcal{H}) < 5a^2 \cdot 2^{-0.03\kappa/a}$, this will follow by piecemeal leakage resilience (Lemma 5.10).
2. In \mathcal{H} , the advantage in distinguishing $(\langle key, \vec{v}_1 \rangle, \langle key, \vec{v}_2 \rangle)$ from uniformly random unbiased bits is bounded by $2^{-0.1\kappa+3}$. I.e., in \mathcal{H} the inner products of \vec{v}_1 and \vec{v}_2 with key are (close to) pairwise independent.

The claim will follow from the two items above (we assume $2^{-0.1\kappa+3} \leq a^2 \cdot 2^{-0.03\kappa/a}$).

Item 1, \mathcal{E} and \mathcal{H} are close. Let \mathcal{A} be an adversary for which we get $\varepsilon = \Delta(\mathcal{E}, \mathcal{H})$. Given \mathcal{A} , we show a piecemeal leakage attack \mathcal{A}' on (key, M) a la Lemma 5.10. We show that if \mathcal{A} has advantage ε in distinguishing \mathcal{E} and \mathcal{H} , then \mathcal{A}' has advantage ε' (where $\varepsilon' \geq \varepsilon \cdot 2^{-a \cdot \lambda}$) in distinguishing whether M is in key 's kernel or M is independent of key . By Lemma 5.10, we conclude a bound on ε' and (through it) on ε .

The piecemeal leakage attack \mathcal{A}' proceeds as follows. The adversary chooses two uniformly random vectors $\vec{v}_1, \vec{v}_2 \in_R \{0, 1\}^\kappa$. It then computes piecemeal leakage $\mathcal{A}(key, (M, \vec{v}_1))$, and also computes whether $\mathcal{A}(key, (M, \vec{v}_1)) = \mathcal{A}(key, (M, \vec{v}_2))$ (for the randomly chosen \vec{v}_1, \vec{v}_2). This requires $(\lambda + 1)$ bits of piecemeal leakage from key and (each piece of) M (it takes λ bits to determine the leakage from each piece \vec{v}_1 and an extra bit to tell whether the leakage on \vec{v}_2 is identical). If the leakage from \vec{v}_1 and \vec{v}_2 is identical, we output

$$\mathcal{A}'(key, M) = (\vec{v}_1, \vec{v}_2, \mathcal{A}(key, (M, \vec{v}_1)))$$

Otherwise, we output $\mathcal{A}'(key, M) = \perp$. Observe now that, conditioning on $\mathcal{A}(key, (M, \vec{v}_1)) = \mathcal{A}(key, (M, \vec{v}_2))$, we have that the output of \mathcal{A}' on M with columns in key 's kernel (together with key) is exactly the distribution \mathcal{E} . The output of \mathcal{A}' on M that is independent of key (conditioned on identical leakage from \vec{v}_1, \vec{v}_2 , and together with key) is distributed exactly as \mathcal{H} . In both cases, when the leakage from \vec{v}_1, \vec{v}_2 is *not* identical, the output is simply \perp . We conclude that the statistical distance ε' between the output of \mathcal{A}' in both cases (M in the kernel and independent M) is at least ε multiplied by the probability that the leakage on \vec{v}_1 and \vec{v}_2 is identical (say w.l.o.g. we refer to the “leakage collision” probability for M in the kernel).

For any fixed (key, M) , the probability that we get identical leakage on \vec{v}_1 and \vec{v}_2 chosen uniformly at random is at least the inverse of the total amount of possible leakage values. I.e. at least $2^{-a \cdot \lambda}$. This gives a lower bound on ε' as a function of ε . By Lemma 5.10 we also have an upper bound on ε' . Putting these together:

$$\varepsilon \cdot 2^{-a \cdot \lambda} \leq \varepsilon' \leq 5a^2 \cdot 2^{-0.04\kappa}$$

we conclude that:

$$\Delta(\mathcal{E}, \mathcal{H}) \leq 5a^2 \cdot 2^{-0.04\kappa} \cdot 2^{a \cdot \lambda} = 5a^2 \cdot 2^{-0.03\kappa}$$

Item 2, \mathcal{H} is pairwise independent. Consider the piecemeal leakage in \mathcal{H} as a multi-source leakage attack on key and on (\vec{v}_1, \vec{v}_2) (chosen conditioned on \vec{v}_1 and \vec{v}_2 yielding the same leakage). For any fixed M , the amount of leakage from key in the attack is bounded by $0.01\kappa/a$. In particular, by Lemma 3.8 we have that, given the leakage, with all but $2^{-0.1\kappa}$ probability, key is an independent sample in a source with min-entropy at least 0.85κ .

We now consider (\vec{v}_1, \vec{v}_2) . We claim that (for any fixed (key, M)) with all but $2^{-0.1\kappa}$ probability over the choice of \vec{v}_1, \vec{v}_2 yielding the same leakage, the set of vectors yielding the same leakage as \vec{v}_1 and \vec{v}_2 is of size at least $2^{0.85\kappa}$. To see this, for a vector \vec{v} , let $S(\vec{v})$ be the set of vectors that give the same leakage as \vec{v} . Let S_{bad} be the set of all vectors \vec{v} for which $S(\vec{v})$ is of size less than $2^{0.85\kappa}$. By Lemma 3.8 we get that:

$$\alpha = \Pr_{\vec{v} \in_R \{0, 1\}^\kappa} [\vec{v} \in S_{bad}] \leq 2^{-0.1\kappa}$$

The probability that \vec{v}_1, \vec{v}_2 drawn s.t. their leakage is identical both land in S_{bad} is at most α^2 divided by the total probability that the leakage from uniformly random \vec{v}_1, \vec{v}_2 is identical (the

“collision probability”). The total leakage is of bounded length $a \cdot \lambda$, so the collision probability is at least $2^{-a \cdot \lambda}$. We conclude that:

$$\Pr_{\vec{v}_1, \vec{v}_2 \in_R \{0,1\}^\kappa: \mathcal{A}(\text{key}, (M, \vec{v}_1)) = \mathcal{A}(\text{key}, (M, \vec{v}_2))} [\vec{v}_1, \vec{v}_2 \in S_{\text{bad}}] \leq \alpha^2 \cdot 2^{a \cdot \lambda} < 2^{-0.1\kappa}$$

We conclude that with all but $2 \cdot 2^{-0.1\kappa}$ probability, given the leakage, the random variables $\text{key}, \vec{v}_1, \vec{v}_2$ are independent and each of min entropy at least 0.85κ . By Lemma 3.7, we conclude that the joint distribution of inner products of \vec{v}_1 and \vec{v}_2 with key is at statistical distance $2^{-0.1\kappa+3}$ from uniformly random (or pairwise independent). ■

5.4 Piecemeal Matrix Multiplication: Security

In this section we use security of random matrices under piecemeal leakage to prove several security properties for piecemeal matrix multiplication. These will serve as building blocks for proving the security of the ciphertext bank as a whole (see the lemmas in Section 5.1). The proofs follow from the lemmas above, and are omitted.

Lemma 5.17. *Take $\kappa, m, n \in \mathbb{N}$ s.t. $m, n \geq \kappa$. Set $\ell = 0.1\kappa$ and leakage bound $\lambda = 0.01\kappa \cdot (\ell/m)^2$. Let \mathcal{A} be any piecemeal adversary and \mathcal{A}' any leakage adversary. Let \mathcal{D} and \mathcal{F} be the following two distributions, where in both cases we draw $\text{key} \in_R \{0,1\}^\kappa$, $\vec{x} \in_R \{0,1\}^m$ and $B \in_R \{0,1\}^{m \times n}$ s.t. the columns of B are all in the kernel of \vec{x} and with parity 1.*

$$\begin{aligned} \mathcal{D} &= (\text{key}, C, w \leftarrow \mathcal{A}_{\kappa, \ell, m, \text{Lin}}^\lambda(\text{key}, \mathbf{A}), \\ &\quad \mathcal{A}'^\lambda(w, \vec{x}, B)[\text{key}, C \leftarrow \text{PiecemealMM}(\mathbf{A}, B)])_{\mathbf{A} \in_R \{0,1\}^{\kappa \times m}: \forall i, \langle \text{key}, \mathbf{A}[i] \rangle = 0} \\ \mathcal{F} &= (\text{key}, C, w \leftarrow \mathcal{A}_{\kappa, \ell, m, \text{Lin}}^\lambda(\text{key}, \mathbf{A}), \\ &\quad \mathcal{A}'^\lambda(w, \vec{x}, B)[\text{key}, C \leftarrow \text{PiecemealMM}(\mathbf{A}, B)])_{\mathbf{A} \in_R \{0,1\}^{\kappa \times n}: \forall i, \langle \text{key}, \mathbf{A}[i] \rangle = \vec{x}[i]} \end{aligned}$$

then $\Delta(\mathcal{D}, \mathcal{F}) = \exp(-\Omega(\kappa))$.

Lemma 5.18. *Take $\kappa, m, n \in \mathbb{N}$ s.t. $m, n \geq \kappa$. Set $\ell = 0.1\kappa$ and leakage bound $\lambda = 0.01\kappa \cdot (\ell/m)^2$. Let \mathcal{A} be any (computationally unbounded) leakage adversary. Let \mathcal{D} and \mathcal{F} be the following two distributions, where in both distributions $\text{key} \in_R \{0,1\}^\kappa$, $\vec{x} \in_R \{0,1\}^{2\kappa}$, $A \in_R \{0,1\}^{\kappa \times m}$ s.t. the i -th column of A has inner product $\vec{x}[i]$ with key :*

$$\begin{aligned} \mathcal{D} &= (\text{key}, A, w \leftarrow \mathcal{A}_{\kappa, \ell, m, \text{Lin}}^\lambda(\text{key}, A), \\ &\quad \mathcal{A}'^\lambda(w)[\text{key}, \vec{c} \leftarrow \text{PiecemealMM}(A, \vec{r})])_{\vec{r} \in_R \{0,1\}^{m \times 1}: (\oplus_i \vec{r}[i]) = 1, \langle \vec{x}, \vec{r} \rangle = 0} \\ \mathcal{F} &= (\text{key}, A, w \leftarrow \mathcal{A}_{\kappa, \ell, m, \text{Lin}}^\lambda(\text{key}, A), \\ &\quad \mathcal{A}'^\lambda(w)[\text{key}, \vec{c} \leftarrow \text{PiecemealMM}(A, \vec{r})])_{\vec{r} \in_R \{0,1\}^{m \times 1}: (\oplus_i \vec{r}[i]) = 1, \langle \vec{x}, \vec{r} \rangle = 1} \end{aligned}$$

then $\Delta(\mathcal{D}, \mathcal{F}) = \exp(-\Omega(\kappa))$.

Lemma 5.19. *Take $\kappa, m, n \in \mathbb{N}$ s.t. $m, n \geq \kappa$. Set $\ell = 0.1\kappa$ and leakage bound $\lambda = 0.01\kappa \cdot (\ell/m)^2$. Let \mathcal{A} be any (computationally unbounded) leakage adversary. Let \mathcal{D} and \mathcal{F} be the following two distributions, where in both distributions $\text{key} \in_R \{0,1\}^\kappa$ and $A \in_R \{0,1\}^{\kappa \times m}$.⁹*

$$\begin{aligned} \mathcal{D} &= (\text{key}, C, \mathcal{A}^\lambda(\text{key}, A)[\text{key}, C \leftarrow \text{PiecemealMM}(A, \mathbf{B})])_{\mathbf{B} \in_R \{0,1\}^{m \times n}: \forall i, \oplus B^T[i] = 1} \\ \mathcal{F} &= (\text{key}, C, \mathcal{A}^\lambda(\text{key}, A)[\text{key}, C \leftarrow \text{PiecemealMM}(A, \mathbf{B})])_{\mathbf{B} \in_R \{0,1\}^{m \times n}: \text{rank}(B) = m-1, \forall i, \oplus B^T[i] = 1} \end{aligned}$$

⁹In both distributions, we give \mathcal{A} complete and explicit access to key and A . The piecemeal leakage attack here is on B , which has different distributions in the two cases.

then $\Delta(\mathcal{D}, \mathcal{F}) = \exp(-\Omega(\kappa))$.

5.5 Ciphertext Bank Security Proofs

We now prove Lemmas 5.1, 5.2, 5.3, 5.4 and 5.5 from Section 5.1 (Lemma 5.1 is the most technically involved). These Lemmas consider leakage produced in an attack on a real or simulated sequence of T ciphertext generations. In proving statistical closeness of the leakage, we will use both the simulated views and additional hybrid views. We compute these views by running the T generations, under the leakage attack of \mathcal{A} , using biased random coins.

Internal Variables. We will use several internal variables as we run these T generations. For the t -th generation (where i goes from 0 to $T-1$): (key_i, C_i) denote the bank before the i -th generation, with underlying plaintexts \vec{x}_i . The randomness used to generate the i -th output ciphertext is \vec{r}_i , the matrix used to refresh the bank is R_i , and the key refresh value is σ_i . We use D_i to denote the intermediate ciphertext in the i -th generation after key refresh, but before multiplication with R_i . The output ciphertext of the i -th generation is \vec{c}_i (the output key is key_i).

Proof of Lemma 5.1. We prove here the case that $b = 0$, the case $b = 1$ is similar. Recall that *Real* is the view of \mathcal{A} given “real” generations of ciphertexts, using a bank of ciphertexts whose underlying plaintexts are 0, and generating ciphertexts whose underlying plaintexts are 0. *Simulated* is a view generated using a bank of ciphertexts whose underlying plaintexts are uniformly random, but choosing plaintexts using biased randomness so that their underlying plaintexts are always 0. The proof of statistical closeness uses a hybrid argument as follows.

We define hybrid views $\{\mathcal{H}_t\}$ for $t \in \{0, \dots, T+1\}$. The output of each hybrid is T tuples, one for each ciphertext generation, each consisting of a key, a ciphertexts, and a leakage value. We compute the hybrids views by running the T generations, under the leakage attack of \mathcal{A} , using biased random coins. We specify the distribution of each of the internal variables described above, and these specify the hybrid view on the outputs and leakage from the T generations.

When generating \mathcal{H}_t , for $t > 0$ we initialize (key_0, C_0) as in *Simulated*. In \mathcal{H}_0 we initialize (key_0, C_0) as in *Real*. We then run T ciphertext generations under \mathcal{A} ’s leakage attack. The key refresh value σ_i is always uniformly random. For the i -th generation, where $i \leq t$, we choose \vec{r}_i uniformly at random s.t. it has odd parity and is in $\text{kernel}(\vec{x}_i)$. For $i \geq t$, we choose \vec{r}_i to be uniformly random with odd parity (and no further restrictions). For $i \neq (t-1)$, we use a uniformly random R_i whose columns have odd parity. For $i = (t-1)$, we use a uniformly random R_i whose columns have odd parity and are in $\text{kernel}(\vec{x}_i)$. This completes the hybrids’ specification.

By construction, we get that $\mathcal{H}_0 = \text{Real}$ and $\mathcal{H}_{T+2} = \text{Simulated}$. It remains to show that, for all t , $\Delta(\mathcal{H}_t, \mathcal{H}_{t+1}) = \exp(-\Omega(\kappa))$. We show this here for $2 < t < T$ (the borderline cases are handled similarly). We use an intermediate distribution \mathcal{H}'_t , which operates as \mathcal{H}_t , except that it chooses a \vec{x}_t vector uniformly at random (recall that in \mathcal{H}_t the columns of C_t are all in $\text{kernel}(key)$). It then chooses \vec{r}_t and the columns of R_t to be uniformly random with odd parity and in $\text{kernel}(\vec{x}_t)$ (whereas in \mathcal{H}_t these were uniformly random with odd parity and no further restriction).

Claim 5.20. $\Delta(\mathcal{H}'_t, \mathcal{H}_{t+1}) = \exp(-\Omega(\kappa))$

Proof. The differences between \mathcal{H}'_t and \mathcal{H}_{t+1} are: (i) the distribution of \vec{r}_{t-1} and the columns of R_{t-1} : they are uniform (with odd parity) in \mathcal{H}_{t+1} , but in \mathcal{H}'_t they are all in $\text{kernel}(\vec{x}_{t-1})$, (ii) in \mathcal{H}'_t we have that the columns of C_t are orthogonal to key_t , whereas in \mathcal{H}_{t+1} they are uniformly

random, and (iii) the distribution of \vec{r}_t and the columns of R_t : they have odd parity in both \mathcal{H}_{t+1} and \mathcal{H}'_t , but in \mathcal{H}_{t+1} they are orthogonal to \vec{x}_t that has the plaintext bits encrypted in C_t , whereas in \mathcal{H}'_t they are orthogonal to a uniformly random \vec{x}_t that is independent of (key_t, C_t) .

We reduce the security game of Lemma 5.17 to distinguishing these two cases. There, a vector \vec{x} is chosen uniformly at random. A matrix A either has columns orthogonal to key , or has uniformly random columns whose inner products with key equal the bits in \vec{x} . This A is multiplied by B with columns in the kernel of \vec{x} . To reduce the game of Lemma 5.17 to distinguishing \mathcal{H}'_t and \mathcal{H}_{t+1} , we put key as key_t , A as C_t , \vec{x} as \vec{x}_t , and B as \mathcal{R}_t .

We begin by showing that leakage from the t -th generation on, together with all keys and ciphertexts created in all generations, is statistically close in both hybrids. The leakage from the t -th generation takes as input the keys and ciphertexts produced in prior iterations, and so for each $i \in \{0, \dots, t-1\}$, we pick (key_i, \vec{c}_i) uniformly at random (independent of (key_t, C_t)) s.t. they have inner product 0. We also choose a uniformly random correlation value σ_t . Note that the distribution of these key-ciphertext pairs, in conjunction with (key_t, C_t) set as above, is exactly as in \mathcal{H}'_t and \mathcal{H}_{t+1} (respectively, depending on the distribution of key and A for the security game of Lemma 5.17).

Using the above reduction, we conclude from Lemma 5.17 that the leakage from the t -th generation, together with $(key_t, \vec{c}_t, \sigma_t, C_{t+1})$ (and the list of key-ciphertext pairs from earlier generations), is statistically close when the random variables are drawn as in \mathcal{H}'_t and \mathcal{H}_{t+1} . We can then use these to generate the leakage and key-ciphertext pairs for generations $(t+1)$ and up (these are just a function of $(key_t, \sigma_t, C_{t+1})$).

We need, however, to also generate the leakage for the ciphertext generations that precede the t -th. Recall that the (key_i, \vec{c}_i) key-ciphertext pairs for all iterations $i < t$ were already chosen and fixed above. We compute the leakage from these iterations using piecemeal leakage from (key_t, C_t) . In fact, for $i \in \{0, \dots, t-3\}$ the leakage is independent of (key_t, C_t) : we simply choose all of the randomness for these generations independently of (key_t, C_t) . For generations $\{0, \dots, t-2\}$, each C_i is sampled uniformly at random. The σ_i values are specified by $key_i \oplus \sigma_i = key_{i+1}$, and these in turn (together with the C_i 's) specify the D_i key-refreshed banks. The R_i matrices are uniformly random s.t. their columns have odd parity and multiplying D_i by R_i yields C_{i+1} . \vec{r}_i 's are uniformly random s.t. they have odd parity and $C_i \times \vec{r}_i = \vec{c}_i$. This completely specifies the randomness for all iterations $i \in \{0, \dots, (t-3)\}$, and we can compute the leakage from those iterations using these values, independently of (key_t, C_t) . Note that the randomness for iterations $t-2$ and $t-1$ will depend on (key_t, C_t) , and so leakage from those iterations is not independent, and will be computed as follows using piecemeal leakage from (key_t, C_t) .

For the $(t-1)$ -th generation, we choose D_{t-1} uniformly at random. The variable σ_{t-1} is a function of key_t (can be accessed via leakage) and of key_{t-1} (which is fixed and public). The ciphertext bank C_{t-1} is a function of D_{t-1} and of σ_{t-1} . I.e. of public information and of (leakage from) key_t . The variable \vec{r}_{t-1} is a function of C_{t-1} and \vec{c}_{t-1} , i.e. of public information and (leakage from) key_t . The only remaining variable which is not specified for iteration $t-1$ is R_{t-1} . We will show below how to compute the needed (piecemeal) leakage from R_{t-1} using D_{t-1} and *piecemeal leakage only* from C_t . Given this (see below), we conclude that leakage from each sub-computation in the $(t-1)$ -th generation can be computed via piecemeal leakage from (key_t, C_t) .

To compute each piece of R_{t-1} used in the piecemeal matrix multiplication, we observe that it suffices to use explicit access to all of D_{t-1} (a “public” uniformly random matrix), together with piecemeal leakage from C_t . We use here the fact that the pieces of R_{t-1} that are needed for

simulating matrix multiplication are all disjoint. Note that, in particular, the distributions of R_{t-1} that we will get in the scenarios of Lemma 5.17 are quite different (as they should be).

Finally, we also need to compute leakage from the $(t-2)$ -th generation. Here we need to specify σ_{t-2} , which is a function of key_{t-2} and key_{t-1} : i.e., we can access is via leakage from key_t . This also specifies D_{t-2} . Finally, for R_{t-2} we use D_{t-2} and C_{t-1} , which can both be accessed via leakage from key_t .

In conclusion, we used a piecemeal attack on (key_t, C_t) to generate the key-ciphertext pairs and leakage up to the t -th generation, and an attack as in Lemma 5.17 to generate the leakage from the t -th generation on. This yielded the views \mathcal{H}'_t and \mathcal{H}_{t+1} . By Lemma 5.17 we conclude that these views must be statistically close. ■

Claim 5.21. $\Delta(\mathcal{H}_t, \mathcal{H}'_t) = \exp(-\Omega(\kappa))$

Proof. The only difference between the hybrids is in the distribution of R_t (in the t -th generation). We reduce the attack game of Lemma 5.19 to distinguishing the two cases. To do so, we generate $(key_i, \vec{c}_i, C_i, w_i)_{i < t}$ identically as in both views. We put $key_{t+1} = key$, $D_t = A$, and $R_t = B$.

Now consider the t -th matrix update in \mathcal{H}_t or \mathcal{H}'_t , performed via piecemeal multiplication of D_t with R_t . In \mathcal{H}_t we have a uniformly random R_t whose columns have odd parity, and in \mathcal{H}'_t we place the additional restriction that the columns are all orthogonal to $kernel(\vec{x}_t)$: i.e. they are all in a (random) subspace of dimension $(2\kappa - 1)$. By Lemma 5.19, the leakage obtained, together with (key_{t+1}, C_{t+1}) , is statistically close in both cases. In both views, we can create the leakage from later rounds as a function of (key_{t+1}, C_{t+1}) (the same function in both cases). We can also create the leakage in the earlier rounds as a function of C_t, key_t as in Claim 5.20 (here this is even easier, because we have explicit access to both). ■

■

Proof of Lemma 5.2. We prove here the case that $b = 0$, the case $b = 1$ is similar. Recall that *Simulated'* and *Simulated''* are two simulated views of \mathcal{A} on a sequence of T simulated generations, both using a bank of ciphertexts whose underlying plaintexts are all uniformly random. The views differ only in that the plaintexts underlying the ciphertexts that are generated, \vec{b}' and \vec{b}'' , might be different. The proof of statistical closeness uses a hybrid argument and follows below.

We define hybrid views $\{\mathcal{H}_t\}$ for $t \in \{0, \dots, T\}$. The output of each hybrid is T leakage values, one for each ciphertext generation. We compute the hybrids views by running the T generations, under the leakage attack of \mathcal{A} , using biased random coins. We will use the internal variables described above as we run these T generations. When generating \mathcal{H}_t , we initialize (key_0, C_0) using *SimBankInit* (so the ciphertexts in the bank are uniformly random). In \mathcal{H}_t , for all i , in the i -th call to *SimBankInit*, we use uniformly random σ_i and R_i whose columns have parity 1. For $i < t$, we choose \vec{r}_i uniformly at random s.t. its parity is 1 and the \vec{c}_i produced has inner product $\vec{b}'[i]$ with key_i . For $i \geq t$, we choose \vec{r}_i similarly, except its inner product with key_i is $\vec{b}''[i]$. This completes the specification of the hybrids.

By construction, we get that $\mathcal{H}_0 = \text{Simulated}''$ and $\mathcal{H}_T = \text{Simulated}'$. It remains to show that, for all t , $\Delta(\mathcal{H}_t, \mathcal{H}_{t+1}) = \exp(-\Omega(\kappa))$. This follows from Lemma 5.18. The Lemma shows that the leakage obtained in the t -th generation, together with (key_{t+1}, C_{t+1}) , is statistically close in \mathcal{H}_t and \mathcal{H}_{t+1} . In both views, we can create the leakage from later rounds as a function of (key_{t+1}, C_{t+1})

(the same function in both cases). We can also create the leakage in the earlier rounds using a piecemeal leakage attack on (key_t, C_t) , as done in Claim 5.20 above. ■

Proof of Lemma 5.3. The distribution \mathcal{D} of $((key_0, \dots, key_{T-1}), (\vec{c}_0, \dots, \vec{c}_{T-1}))$ in *Simulated* (without any leakage) is IuO, with orthogonality \vec{b} and underlying distributions \mathcal{K} and \mathcal{C} on keys and ciphertexts: each key and ciphertext in the underlying distributions is uniformly and independently random. Now, observe that the ciphertext banks in *Simulated* are uniformly random, independent of the keys and ciphertexts. Thus, we can compute the leakage from all T generations as a (randomized) multi-source function operating on the ciphertext banks and separately on the keys and separately on each ciphertext. We conclude by Lemma 3.11 that for each i the distribution $\mathcal{D}_i(w)$ is indeed IuO, with orthogonality $\vec{b}[i]$ and with underlying distributions $\mathcal{K}_i(w)$ and $\mathcal{C}_i(w)$ that do not depend on $\vec{b}[i]$. The entropy bounds on each key and ciphertext (given w) follow by Lemma 3.8.

We note that independence up to orthogonality and high entropy hold even given the explicit lists of ciphertexts in the bank (in all calls), as these are just uniformly random matrices, and even given the random coins used to compute leakage from the target generations (given the ciphertext bank and the target ciphertext). ■

6 Safe Computations

In this section we present the *SafeNAND* procedure, see Section 2.2 for an overview. The (simpler) treatment of duplications gates is omitted.

This section is organized as follows: the *SafeNAND* procedure and its security properties are in Section 6.1. This procedure uses a leakage-resilient permutation procedure, *Permute*, which is presented and proved secure in Section 6.2. We then use *Permute*'s security in the proof of *SafeNAND*'s security, which follows in Section 6.3.

6.1 Safe Computations: Interface and Security

In this section we present the procedure for safely computing NAND gates. The full procedure is in Figure 8. Correctness follows from the description (see the introduction). For security, we show that a view of the NAND computation can be simulated, given only the output (and the underlying distributions of the input keys and the input ciphertexts). This is formalized in Lemma 6.1. See the subsequent sections for details of the *Permute* procedure and the proof of Lemma 6.1.

Security of *SafeNAND*. We provide a simulator for producing the leakage on the *SafeNAND* procedure, when the inputs to *SafeNAND* are chosen from an IuO distribution. The simulator is given a_k , and the underlying distributions for the which the *SafeNAND* inputs were drawn. It outputs a complete view of the leakage from *SafeNAND*. This includes the leakage from the *Decrypt* operation (which loads keys and ciphertexts into memory simultaneously). The security claim is below in Lemma 6.1. We note that the *SafeNAND* simulator is not efficient, its running time might be exponential in that of the leakage adversary. The descriptions of the underlying input distributions themselves might already be of exponential size. This does not pose a problem, because the security of our main construction is statistical, and we never use the *SafeNAND*

$\text{SafeNAND}(a_i, \text{key}_i, \vec{c}_i, a_j, \text{key}_j, \vec{c}_j, \text{key}_k, \vec{c}_k)$: Safe NAND computation

1. Correlate the ciphertexts to a new key. Pick a new key $\text{key} \leftarrow \text{KeyGen}(1^\kappa)$
 $\sigma_i \leftarrow \text{key}_i \oplus \text{key}, \sigma_j \leftarrow \text{key}_j \oplus \text{key}, \sigma_k \leftarrow \text{key}_k \oplus \text{key}$
 $c'_i \leftarrow \text{CipherCorrelate}(c_i, \sigma_i), c'_j \leftarrow \text{CipherCorrelate}(c_j, \sigma_j), c'_k \leftarrow \text{CipherCorrelate}(c_k, \sigma_k)$
leakage on $[(\text{key}_i, \sigma_i), (\text{key}_j, \sigma_j), (\text{key}_k, \sigma_k), (\vec{c}_i, \sigma_i), (\vec{c}_j, \sigma_j), (\vec{c}_k, \sigma_k)]$
2. $c''_i \leftarrow c'_i \oplus (a_i, 0, \dots, 0), c''_j \leftarrow c'_j \oplus (a_j, 0, \dots, 0)$
 $C \leftarrow (\vec{c}_k, \vec{c}_k \oplus \vec{c}'_i, \vec{c}'_j, \vec{c}_k \oplus \vec{c}'_j, \vec{c}'_i \oplus \vec{c}'_j \oplus (1, 0, \dots, 0))$
leakage on ciphertexts
3. $(K', C') \leftarrow \text{Permute}(\text{key}, C)$
leakage from Permute (see below)
4. Decrypt the four ciphertexts in C' using the four keys in K' . If there is one 0 plaintext in the results, then output $a_k \leftarrow 0$. Otherwise, output $a_k \leftarrow 1$
leakage on C' and K' (jointly)

Figure 8: *SafeNAND* procedure. The *Permute* procedure is in Figure 9.

simulator for the (efficient) *SimEval* simulation procedure, only for creating hybrid distributions in the security proof.

Lemma 6.1. *There exist: an exponential time simulator SimNAND , a leakage bound $\lambda(\kappa) = \tilde{\Omega}(\kappa)$, and a distance bound $\delta(\kappa) = \text{negl}(\kappa)$ s.t. for every $\kappa \in \mathbb{N}$ and leakage adversary \mathcal{A} :*

*Let \mathcal{D} be a distribution on two 3-tuples: a key-tuple $(\text{key}_i, \text{key}_j, \text{key}_k) \in \{0, 1\}^{3 \times \kappa}$, and a ciphertext-tuple $(\vec{c}_i, \vec{c}_j, \vec{c}_k) \in \{0, 1\}^{3 \times \kappa}$. Suppose that \mathcal{D} is *IuO* with orthogonality $(b_i, b_j, b_k) \in \{0, 1\}^3$. Let \mathcal{D} 's underlying distributions on the key-tuple and on the ciphertext-tuple be \mathcal{K} and \mathcal{C} . I.e. $\mathcal{D} = \mathcal{K} \perp_{(b_i, b_j, b_k)} \mathcal{C}$. Suppose further that $H_\infty(\mathcal{K}), H_\infty(\mathcal{C}) \geq 3\kappa - O(\lambda(\kappa))$.*

For any $(a_i, a_j) \in \{0, 1\}^2$, take:

$$\begin{aligned} \text{Real} &= \left(\mathcal{A}^{\lambda(\kappa)}[a_k \leftarrow \text{SafeNAND}(a_i, \text{key}_i, \vec{c}_i, a_j, \text{key}_j, \vec{c}_j, \text{key}_k, \vec{c}_k)] \right)_{((\text{key}_i, \text{key}_j, \text{key}_k), (\vec{c}_i, \vec{c}_j, \vec{c}_k)) \sim \mathcal{D}} \\ \text{Simulated} &= (\text{SimNAND}(a_i, a_j, a_k, \mathcal{K}, \mathcal{C}))_{a_k \leftarrow ((a_i \oplus b_i) \text{ NAND } (a_j \oplus b_j)) \oplus b_k} \end{aligned}$$

then $\Delta(\text{Real}, \text{Simulated}) \leq \delta(\kappa)$.

6.2 Leakage-Resilient Permutation

The *Permute* procedure receives as input a key and a 4-tuple of ciphertexts. It outputs a “fresh” pair of 4-tuples of keys and ciphertexts. The correctness property of the permute procedure is that the plaintexts underlying the output ciphertexts (under the respective output keys) are a (random) permutation of the plaintexts underlying the input ciphertexts. The intuitive security guarantee is that, even to a computationally unbounded leakage adversary, the permutation looks uniformly random. The procedure is below in Figure 9. Correctness is immediate. Security is formalized by the existence of a simulator that generates a complete view of the leakage *and the output keys and ciphertexts*. The simulator only gets: (i) descriptions of the marginal distribution from which *key* and the input ciphertext are drawn, and (ii) a random permutation of the plaintexts underlying the

input ciphertexts. We show that, under the appropriate conditions on the distribution from which the key and ciphertexts are drawn, the real and simulated joint distributions of leakage and output from *Permute* will be statistically close. In particular, on an intuitive level, the joint distribution of the leakage and the outputs is independent of the permutation that was used. This security property is stated in Lemma 6.2 below. We note that the simulator is not efficient, and may run in exponential time (as was the case for the *SimNAND* simulator, it is only used in the security proof of our main construction).

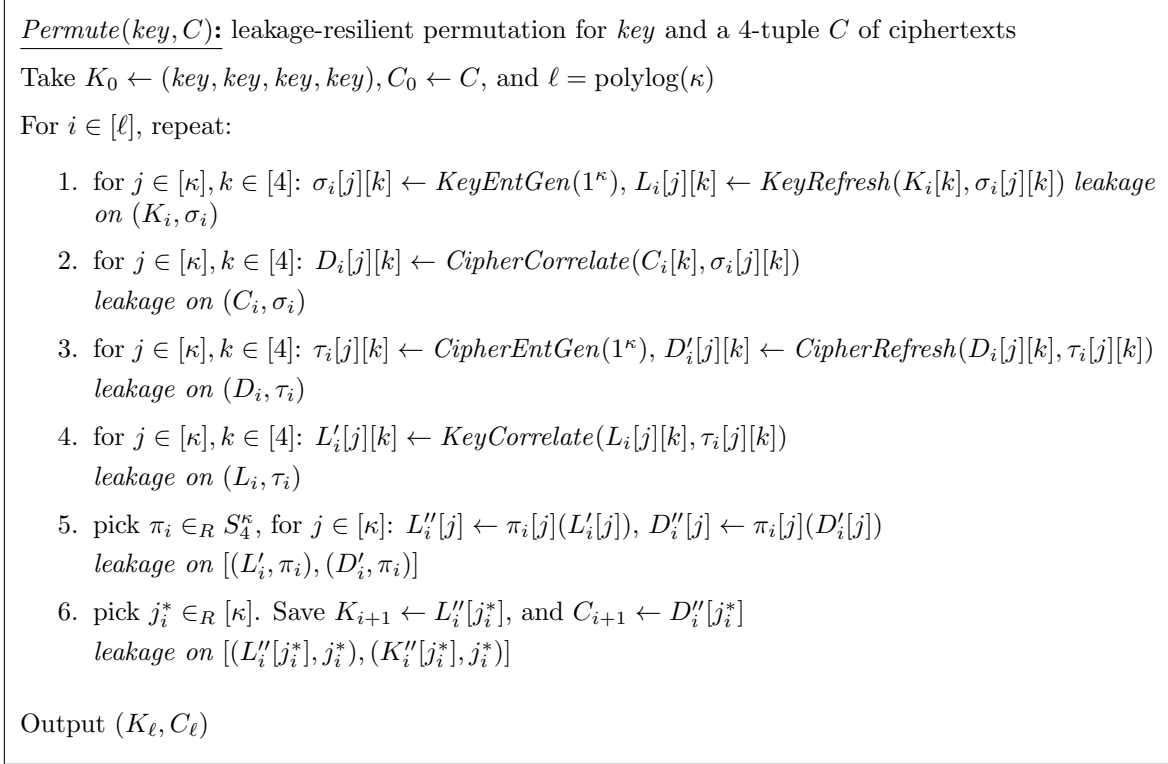


Figure 9: Leakage-Resilient Ciphertext Permutation for $\kappa \in \mathbb{N}$

Lemma 6.2. *There exists an exponential-time simulator $\textit{SimPermute}$, a leakage bound $\lambda(\kappa) = \tilde{\Omega}(\kappa)$, and a distance bound $\delta(\kappa) = \text{negl}(\kappa)$, s.t for any $\kappa \in \mathbb{N}$ and leakage adversary \mathcal{A} :*

Let \mathcal{D} be a distribution on $\textit{key} \in \{0, 1\}^\kappa$ and a ciphertext 4-tuple $C \in \{0, 1\}^{4 \times \kappa}$. Suppose that \mathcal{D} is IuO with orthogonality $\vec{b} \in \{0, 1\}^4$. Let \mathcal{K} and \mathcal{C} be \mathcal{D} 's underlying distributions on key and on C . Suppose further that $H_\infty(\mathcal{K}) \geq \kappa - O(\lambda(\kappa))$ and $H_\infty(\mathcal{C}) \geq 3\kappa - O(\lambda(\kappa))$.

Take Real and Simulated to be the following views:

$$\begin{aligned} \textit{Real} &= \left(K', C', \mathcal{A}^{\lambda(\kappa)}[(K', C') \leftarrow \textit{Permute}(\textit{key}, C)] \right)_{(\textit{key}, C) \sim \mathcal{D}} \\ \textit{Simulated} &= \left(\textit{SimPermute}(\vec{b}', \mathcal{K}, \mathcal{C}) \right)_{\mu \in_R S_4, \vec{b}' \leftarrow \mu(\vec{b})} \end{aligned}$$

then $\Delta(\textit{Real}, \textit{Simulated}) \leq \delta(\kappa)$.

Proof. We begin by describing the *SimPermute* Simulator. The proof that *Real* and *Simulated* are statistically close follows.

SimPermute. The simulator samples $key \sim \mathcal{K}$ and $C \sim \mathcal{C}$, conditioned on the inner product of key with C being $\vec{0}$ (rather than $\vec{b}[i]$, as in \mathcal{D}). The simulator then runs *Permute* on (key, C) , under \mathcal{A} 's leakage attack, to compute the leakage w . To compute the output (K', C') , the simulator first samples an input key and randomness \vec{r} for *Permute*, from the conditional underlying distribution of key and the randomness given leakage w . Note that, as in Lemma 3.11, this conditional distribution depends only on \mathcal{K} (and not on \mathcal{C}). Using key and \vec{r} , the simulator can compute the K' that *Permute* would output. Similarly, the simulator computes the conditional distribution of C' , given w and \vec{r} . Again, as in Lemma 3.11, this depends only on \mathcal{C} (and not on \mathcal{K}). The simulator samples C' from this conditional distribution, under the additional condition that the inner products of C' with K' equal \vec{b}' . The output is (K', C', w) .

Statistical Closeness of *Real* and *Simulated*. We first observe that w is \mathcal{A} 's output in a leakage attack that operates separately on key and on C . Moreover, the leakage on key and on C is of bounded total length $O(\ell \cdot \lambda(\kappa)) \ll \kappa$. Since the “real” distribution \mathcal{D} of (key, C) is IuO, by Lemma 3.11 the distributions of w in *Real* and in *Simulated* are $\delta(\kappa)$ -statistically close.

The more difficult part of the proof is arguing that (w.h.p. over w), the distributions of (K', C') , conditioned on w , in *Real* and in *Simulated* are statistically close. For this, we consider a hybrid distribution *Real'*. To generate *Real'*, we compute w as in *Simulated*, by running \mathcal{A} 's leakage attack on *Permute*, activated on key and C chosen s.t. their inner products equal $\vec{0}$. Let π be the composition of the permutations chosen in the ℓ iterations of *Permute*. In *Real'* we generate (K', C') as in *Simulated*, but conditioning the underlying output distributions on the inner products of K' and C' equalling $\vec{b}' = \pi(\vec{b})$, rather than \vec{b}' which is a uniformly random permutation of \vec{b} in *Simulated*. We show that *Real'* is statistically close to both *Real* and *Simulated*.

Proposition 6.3. $\Delta(\textit{Real}, \textit{Real}') = O(\delta(\kappa))$

Proof. We re-cast *Real* by considering the following procedure for generating it. This alternate generation operates as *Real*, except that when drawing the input (key, C) from the underlying distributions, we condition on the inner products equalling \vec{b} (rather than $\vec{0}$). These (key, C) are used to compute w , and then (K', C') are drawn (as in *Real'*) from their conditional distributions, conditioned on inner products $\pi(\vec{b})$ (where π is the composition of the permutations used in all ℓ iterations of *Permute*). Since the input distribution \mathcal{D} used in *Real* is IuO, this procedure generates exactly the view *Real*.

We now show that *Real* and *Real'* are statistically close. These two distributions differ only in the joint distribution of (w, π) ; given w and π , the distributions of (K', C') derived in *Real* and *Real'* are identical. (w, π) are generated via a multi-source leakage attack, operating separately on key and on C , with a total of $O(\ell \cdot \lambda(\kappa)) \ll \kappa$ bits of leakage. Moreover, the distributions of key and C in *Real* and *Real'* are both IuO (by construction), and differ only in their orthogonalities (\vec{b} or $\vec{0}$ respectively). By Lemma 3.11, we get that the distributions of (w, π) in *Real* and *Real'* are $\delta(\kappa)$ -statistically close, and thus so are *Real* and *Real'* themselves. ■

Proposition 6.4. $\Delta(\textit{Real}', \textit{Simulated}) = \text{negl}(\kappa)$

Proof. Recall that the distributions of w in *Real'* and *Simulated* are identical. The underlying distributions on K' and on C' (given w) are also identical. The difference is in the distribution (given w) of the permutation used to compute \vec{b}' from the given vector \vec{b} (\vec{b}' is then used to jointly sample (K', C')). In *Real'* we have $\vec{b}' = \pi(\vec{b})$, where π is the composition of permutations chosen

by *Permute* in its ℓ iterations. In *Simulated* we have $\vec{b}' = \mu(\vec{b})$, where μ is a uniformly random permutation in S_4 , independent of w . We will show that, for *any* input (key, C) , the distribution of π in *Real'* (conditioned on w), is $\text{negl}(\kappa)$ -close to uniformly random (w.h.p over the leakage w). It follows that *Real'* and *Simulated* are $\text{negl}(\kappa)$ -statistically close.

The intuition, loosely speaking, is that for each $i \in [\ell]$, the permutation $\pi_i^* = \pi_i[j_i^*]$ chosen in *Permute*'s i -th iteration, looks “fairly random” even given w . Moreover, these ℓ permutations are drawn independently from their “fairly random” distributions. The composition, over all ℓ iterations of *Permute*, of the permutations chosen in each iteration, is thus statistically close to uniformly random. We formalize this intuition below, starting with the notion of “well-mixing” distributions over in S_4 .

Definition 6.5 (Well-Mixing Distribution on Permutations). A distribution P over S_4 is said to be *well-mixing* if:

$$H_\infty(P) \geq 0.99 \log |S_4|$$

Next, we observe that the composition of a sequence of permutations drawn from well-mixing distributions is itself very close to uniform.

Claim 6.6. For any sequence $P_0, \dots, P_{\ell-1}$ of well-mixing distributions, let P be:

$$P \triangleq (\pi_0 \circ \dots \circ \pi_{\ell-1})_{\pi_0 \sim P_1, \dots, \pi_{\ell-1} \sim P_{\ell-1}}$$

then P is $\exp(-\Omega(\ell))$ -close to uniform over S_4 .

For *Permute*'s i -th iteration, let w_i be the leakage in that iteration. We define P_i to be the distribution of the permutation $\pi_i^* = \pi_i[j_i^*]$ chosen in the i -th iteration, conditioned on (w_0, \dots, w_i) and also on the keys and ciphertexts $(K_i, C_i, K_{i+1}, C_{i+1})$. We show that in *Real'*, with overwhelming probability over the random coins up to (but not including) the choice of j_i^* , with probability at least $1/2$ over *Permute*'s choice of j_i^* , the distribution P_i is well-mixing.

Claim 6.7. For the view *Real'*, for any $i \in [\ell]$, and for any $(K_i, C_i, (w_0, \dots, w_{i-1}))$, with all but $O(\delta(\kappa))$ probability over *Permute*'s random choices in iteration i up to Step 6, with probability at least $1/2$ over *Permute*'s choice of j_i^* in Step 6, the distribution P_i is well-mixing.

Proof. Examine the distribution of the vector π_i of permutations used in iteration i , conditioned on $(K_i, C_i, (w_0, \dots, w_{i-1}))$, and conditioned also on (L_i'', D_i'') (but without conditioning on the leakage w_i in the i -th iteration or on j_i^*). Here the randomness is over $(\sigma_i, \tau_i, \pi_i)$. We observe that in this conditional distribution, the marginal distribution on $(\pi_i[0], \dots, \pi_i[\kappa-1])$ is uniformly random over S_4^κ . This is because for each $j \in [\kappa]$, the pair $(\sigma_i[j], \tau_i[j])$ are uniformly random (under the condition that they maintain the underlying 0 plaintext bits in \vec{b}_i). Thus, $\sigma_i[j], \tau_i[j]$ completely “mask” the permutation $\pi_i[j]$ that was used: all permutations are equally likely. Note that here we use the fact that the plaintext bits \vec{b}_i underlying (K_i, C_i) in *Real'* are all identical (they all equal 0). Otherwise, since *Permute* preserves the set of underlying plaintexts (if not their order), there would be information about each $\pi_i[j]$ in the plaintexts underlying (L_i'', D_i'') .

By Lemma 3.8, since the leakage w_i on $(\sigma_i, \tau_i, \pi_i)$ is of length at most $O(\lambda(\kappa))$ bits, with all but $\delta(\kappa)$ probability, the min-entropy of the vector π_i given $(K_i, C_i, L_i'', D_i'', (w_0, \dots, w_{i-1}, w_i))$ is at least $0.995 \cdot \kappa \cdot \log |S_4|$. By an averaging argument, with probability at least $1/2$ over *Permute*'s (uniformly random) choice of j_i^* , we get that the min entropy of $\pi_i^* = \pi_i[j_i^*]$, given $(K_i, C_i, L_i'', D_i'', (w_0, \dots, w_{i-1}, w_i))$, is at least $0.99 \log |S_4|$. The claim about P_i follows (in P_i we

condition π_i^* on the same information as above, except we replace (L_i'', D_i'') with just $(K_{i+1}, C_{i+1}) = (L_i''[j_i^*], D_i''[j_i^*])$. ■

To complete the proof of Proposition 6.4, we examine the composed distribution $(\pi = (\pi_0^* \circ \dots \circ \pi_{\ell-1}^*)|w)$. Each π_i^* is drawn from P_i , and these draws are all independent of each other. By Claim 6.7, we get that with all but $\exp(-\Omega(\ell))$ probability over the random coins, fixing the sequence $((K_0, C_0), \dots, (K_{\ell-1}, C_{\ell-1}))$ and the leakage w , at least $1/3$ of the distributions P_i are well-mixing. When this happens, by Claim 6.6, the distribution of $(\pi|w)$ is $\exp(-\Omega(\ell))$ -close to uniform, where $\ell = \text{polylog}(\kappa)$. ■

■

6.3 Proof of *SimNAND* Security (Lemma 6.1)

Remark 6.8. We will assume throughout this section that the leakage w from *SafeNAND* includes *Permute*'s output in its entirety. This is a strengthening of the leakage adversary (it gets more leakage “for free”), and so it strengthens our security claim for *SafeNAND*.

Proof of Lemma 6.1. We begin by describing the *SimNAND* simulator, and then proceed with a proof of statistical closeness of *Real* and *Simulated*.

SimNAND. Let \mathcal{D}^\times be the independently drawn variant of \mathcal{D} (as in Definition 3.10, i.e. with independent draws from \mathcal{K} and from \mathcal{C}). *SimNAND* samples $((key_i, key_j, key_k), (\vec{c}_i, \vec{c}_j, \vec{c}_k)) \sim \mathcal{D}^\times$, and $key \leftarrow \text{KeyGen}(1^\kappa)$. It runs Steps 1 and 2 of the *SafeNAND* procedure, on the keys and ciphertexts it drew, under \mathcal{A} 's leakage attack. Let $w_{1,2}$ be the leakage generated in this partial execution, and let $\vec{\sigma} = (\sigma_i, \sigma_j, \sigma_k)$ be the correlation values computed by *SafeNAND* in this simulated execution.

Next, *SimNAND* computes $\mathcal{K}_{\text{SimPermute}}$ and $\mathcal{C}_{\text{SimPermute}}$, the conditional distributions of key and of C in Step 3, given $w_{1,2}$ and $\vec{\sigma}$. *SimNAND* proceeds to simulate Step 3 by calling the *Permute* simulator, *SimPermute* (see Lemma 6.2), on input $(\vec{b}_{\text{SimPermute}}, \mathcal{K}_{\text{SimPermute}}, \mathcal{C}_{\text{SimPermute}})$, where $\vec{b}_{\text{SimPermute}}$ is a uniformly random permutation of the vector $(a_k, a_k \oplus 1, a_k \oplus 1, a_k \oplus 1)$. *SimPermute*'s output includes the leakage w_3 from Step 3 and an output (K', C') from *Permute*. *SimNAND* completes the simulation by running Step 4 on (K', C') under \mathcal{A} 's leakage attack, producing leakage w_4 . The leakage that *SimNAND* outputs is the accumulated leakage $w = (w_{1,2} \circ w_3 \circ (K', C') \circ w_4)$ from all the simulated steps of *SafeNAND* (recall from Remark 6.8 that we include (K', C') in the leakage).

Statistical closeness of *Real* and *Simulated*. We examine the distributions of the leakage $w_{1,2}$ in Steps 1 and 2 in both views. In both *Real* and *Simulated* we have $(key_i, key_j, key_k) \sim \mathcal{K}$ and $key \leftarrow \text{KeyGen}(1^\kappa)$. These determine the correlation values $\vec{\sigma} = (\sigma_i, \sigma_j, \sigma_k)$ computed in Step 1 of *SafeNAND*. Note that the correlation values are a function of the keys only (and not the ciphertexts), and thus they are identically distributed in both *Real* and *Simulated*. The difference is in the conditional distribution of $(\vec{c}_i, \vec{c}_j, \vec{c}_k)$ given $(key, \vec{\sigma})$.

We focus on the joint conditional distribution of $(key, (\vec{c}_i, \vec{c}_j, \vec{c}_k))$, conditioned on $\vec{\sigma}$. We will show that this joint distribution, conditioned on $\vec{\sigma}$, is: (i) *IuO* in *Real*, and (ii) its independently drawn variant in *Simulated*. Given $\vec{\sigma}$, the leakage is a multi-source function of key and of the ciphertexts. We will conclude, using Lemma 3.11, that: (i) the leakage in *Real* and in *Simulated*

is statistically close, and (ii) for a fixed leakage value $w_{1,2}$ that can occur in *Real*, the conditional distribution of $(key, (\vec{c}_i, \vec{c}_j, \vec{c}_k))$ in *Real*, given $(\vec{\sigma}, w_{1,2})$, will remain IuO. The distribution of $(key, (\vec{c}_i, \vec{c}_j, \vec{c}_k))$ in *Simulated*, given $(\vec{\sigma}, w_{1,2})$, will be the independently drawn variant of the same distribution in *Real*. Note that C is a (linear) function of $(\vec{c}_i, \vec{c}_j, \vec{c}_k)$ (and a_i, a_j), and so we get the same guarantees for the distributions of (key, C) in *Real* and *Simulated*.

It remains to show that the requirements of Lemma 3.11 hold. Namely, that $(key, (\vec{c}_i, \vec{c}_j, \vec{c}_k))$ in *Real*, conditioned on $\vec{\sigma}$, is IuO, and that the same distribution in *Simulated* is its independently drawn variant. For this, observe first that in *Simulated*, the distribution of $(key, (\vec{c}_i, \vec{c}_j, \vec{c}_k))$ given $\vec{\sigma}$ is the product distribution of key given $\vec{\sigma}$, and of \mathcal{C} (without any conditioning). The fixed values of $\vec{\sigma}$ do not effect the marginal distribution on ciphertexts, because (in *Simulated*) the ciphertexts are drawn independently of the keys. In *Real*, on the other hand, the keys and ciphertexts are no longer drawn independently. However, even in *Real*, \mathcal{D} is IuO. In particular, \mathcal{D} 's marginal conditional distribution on $(\vec{c}_i, \vec{c}_j, \vec{c}_k)$, given key and $\vec{\sigma}$, is equal to \mathcal{C} , conditioned on $\langle key \oplus \sigma_i, \vec{c}_i \rangle = b_i$, on $\langle key \oplus \sigma_j, \vec{c}_j \rangle = b_j$, and on $\langle key \oplus \sigma_k, \vec{c}_k \rangle = b_k$. We conclude that in *Real*, the distribution of $(key, (\vec{c}_i, \vec{c}_j, \vec{c}_k))$, conditioned on $\vec{\sigma}$, is also IuO. Moreover, by Lemma 3.8, with all but $O(\delta(\kappa))$ probability over $\vec{\sigma}$, the min-entropy of $(key, (\vec{c}_i, \vec{c}_j, \vec{c}_k))$ given $\vec{\sigma}$ is at least $4\kappa - O(\lambda(\kappa))$.

By Lemma 3.11, we conclude that the distributions $((key, C)|(\vec{\sigma}, w_{1,2}))$ in *Real* and in *Simulated* satisfy all the conditions of Lemma 6.2 (security of *Permute*). By construction, the vector $\vec{b}_{SimPermute}$ of plaintext values given as input to the *SimPermute* simulator in *Simulated*, is a uniformly random permutation of the plaintexts underlying (key, C) , the input to *Permute* in *Real*. By Lemma 6.2, we conclude that the distributions of $(w_{1,2} \circ w_3 \circ (K', C'))$, in conjunction with $\vec{\sigma}$, are statistically close in *Real* and *Simulated*. Statistical closeness of *Real* and *Simulated* follows, because the leakage w_4 from Step 4 is a function of (K', C') . ■

7 Putting it Together: The Full Construction

In this section we show how to compile any circuit into a secure transformed one that resists OC side-channel attacks, as per Definition 3.14 in Section 3.4. See Section 2 for an overview of the construction.

The full initialization and evaluation procedures are presented below in Figures 10 and 11. The evaluation procedure is separated into sub-computations (which may themselves be separated into sub-computations of the cryptographic algorithms). Ciphertext bank procedures are in Section 5. The procedures for safely computing NAND and duplication are in Section 6. Theorem 7.1 states the security of the compiler.

Theorem 7.1. *There exist a leakage bound $\lambda(\kappa) = \tilde{\Omega}(\kappa)$ and a distance bound $\delta(\kappa) = \text{negl}(\kappa)$, s.t. for every $\kappa \in \mathbb{N}$, the $(Init, Eval)$ compiler specified in Figures 10 and 11 is a (λ, δ) -continuous leakage secure compiler, as per Definition 3.14.*

Proof Sketch. We first specify the simulator and then provide a sketch of statistical security.

Simulator. Let \mathcal{A} be a (continuous) leakage adversary. The simulator, using *SimInit* and *SimEval*, creates a view of repeated executions of *Eval*, on different inputs, under a (continuous) leakage attack by \mathcal{A} . It mimics the operation of the “real” *Eval* procedure. The *SimInit* procedure starts by initializing all ciphertext banks using *SimBankInit*. Within the t -th execution,

Initialization $Init(1^\kappa, C, y)$

1. for every y -input wire i , corresponding to $y[j]$:

$$Bank_i \leftarrow BankInit(1^\kappa, y[j])$$

2. for the output wire $output$:

$$Bank_{output} \leftarrow BankInit(1^\kappa, 0)$$

3. for the internal wires:

$$Bank_{internal} \leftarrow BankInit(1^\kappa, b), \text{ where } b \in_R \{0, 1\}$$

4. output: $state_0 \leftarrow (Bank_{internal}, Bank_{output}, \{Bank_i\}_i \text{ is a } y\text{-input wire})$

Figure 10: *Init* procedure, to be run in an offline stage on circuit C and secret y .

with input x_t and output $C(y, x_t)$, the simulator picks all of the (a_i, b_i) shares for each wire i in advance. To do so, the simulator first evaluates $C(\vec{0}, x_t)$ and takes v'_i to be the bit value on wire i in this evaluation. For y -input wires, the simulator sets $a_i = b_i = 0$. For internal wires, the a_i shares are uniformly random, and each b_i is set so that $a_i \oplus b_i = v'_i$. For the output wire out , the simulator sets $a_{out} = C(y, x_t)$, and $b_{out} = v'_{out} \oplus a_{out}$.

Once the (a_i, b_i) values are picked, the simulator generates the ciphertexts \vec{c}_i so that the plaintext underlying \vec{c}_i is indeed b_i . This is done using the *SimBankGen* simulation procedures, which gives the simulator control over the plaintext underlying the ciphertext that it generates. The rest of the simulator's operation follows the *Eval* procedure on the generated ciphertexts, and the leakage is generated as it would be from an execution of *Eval*. The *SimInit* and *SimEval* procedures are specified below in Figures 12 and 13.

Statistical Security (Sketch). The intuition for security is that the “public” a_i shares in the simulated execution are distributed exactly as they are in the real execution. The “private” b_i shares differ between the real and simulated execution, but these shares are in protected LROTP encrypted form (key_i, \vec{c}_i) , where the key and ciphertext are never loaded into memory together.

The full proof that *Real* and *Simulated* are statistically close uses several hybrids:

Real to HybridReal: replacing real generations with simulated ones. The first hybrid is *HybridReal*. It is obtained from *Real* by replacing each “real” generation with a “simulated” generation that produces a key-ciphertext pair with the same underlying plaintext. In particular, we replace each *BankInit*(b_i) call for an output or y -input wire i , with a *SimBankInit* call, and we replace the *BankInit* call for $Bank_{internal}$ with a *SimBankInit* call. We then replace each *BankGen* call for an output or y -input wire i with a call to *SimBankGen*(b_i), where b_i is the appropriate private share for wire i . We replace each pair of *BankGen* calls to $Bank_{internal}$ with a pair of calls to *SimBankGen*(b), where b is independent and uniformly random in $\{0, 1\}$. Finally, we replace each call to *BankUpdate* and *BankRedraw* with a call to *SimBankUpdate* and *SimBankRedraw* (respectively). Other than these changes to the ciphertext bank calls, we run exactly as in *Real*.

Evaluation $Eval(state_{t-1}, x_t)$

$state_{t-1} = (Bank_{internal}, Bank_{output}, \{Bank_i\}_i \text{ is a } y\text{-input wire})$

1. Generate keys and ciphertexts for all circuit wires:

(a) y input wire i :

$$(key_i, \bar{c}_i^{in}) \leftarrow BankGen(Bank_i)$$

$$Bank_i \leftarrow BankUpdate(Bank_i)$$

(b) output wire $output$:

$$(key_{output}, \bar{c}_{output}^{out}) \leftarrow BankGen(Bank_{output})$$

$$Bank_{output} \leftarrow BankUpdate(Bank_{output})$$

(c) each internal wire i (in sequence):

$$(key_i, \bar{c}_i^{in}) \leftarrow BankGen(Bank_{internal})$$

$$(key_i, \bar{c}_i^{out}) \leftarrow BankGen(Bank_{internal})$$

$$Bank_{internal} \leftarrow BankRedraw(Bank_{internal})$$

$$Bank_{internal} \leftarrow BankUpdate(Bank_{internal})$$

(d) x_t -input wire i : $key_i \leftarrow KeyGen(1^\kappa), \bar{c}_i^{in} \leftarrow Encrypt(key_i, 0)$

2. Compute the public shares on all wires.

For the input wires: for each y -input wire i , $a_i \leftarrow 0$. For each x -input wire i corresponding to $x_t[j]$, $a_i \leftarrow x_t[j]$.

Proceed layer by layer (from input to output) to compute the remaining public shares:

(a) for each NAND gate with input wires i, j and output wire k , compute:

$$a_k \leftarrow SafeNAND(a_i, key_i, \bar{c}_i^{in}, a_j, key_j, \bar{c}_j^{in}, key_k, \bar{c}_k^{out})$$

(b) for each duplication gate with input wire i and output wires j, k , compute:

$$a_j \leftarrow SafeDup(a_i, key_i, \bar{c}_i^{in}, key_j, \bar{c}_j^{out})$$

$$a_k \leftarrow SafeDup(a_i, key_i, \bar{c}_i^{in}, key_k, \bar{c}_k^{out})$$

(c) output a_{output}

3. the new state is: $state_t \leftarrow (Bank_{internal}, Bank_{output}, \{Bank_i\}_i \text{ is a } y\text{-input wire})$

Figure 11: *Eval* procedure performed on input x_t , under OC leakage. See Section 5.1 for ciphertext bank procedures, Section 6.1 for *SafeNAND*, *SafeDup*.

The two views *Real* and *HybridReal* differ only in that in *Real* we have calls to *BankInit*, *BankGen*, *BankUpdate*, *BankRedraw*, whereas in *HybridReal* we have calls to the corresponding simulated procedures. Note that the b_i values given as input to *SimBankGen* in *HybridReal* are distributed *identically* to the plaintexts underlying the ciphertexts generated in the corresponding calls to *BankGen* in *Real*: for y -input wire i , corresponding to the j -th bit of y , b_i is equal to $y[j]$ in both views. For each internal wire i , b_i is an independently uniformly random bit in both views. For the output wire $output$, b_{output} equals 0 in both views. By Lemmas 5.1 and 5.4, we get that the

Simulator Initialization $SimInit(1^\kappa, C)$

1. for every y -input wire i , corresponding to $y[j]$:

$$Bank_i \leftarrow SimBankInit(1^\kappa)$$

2. for the output wire $output$:

$$Bank_{output} \leftarrow SimBankInit(1^\kappa)$$

3. for the internal wires:

$$Bank_{internal} \leftarrow SimBankInit(1^\kappa)$$

4. output: $state_0 \leftarrow (Bank_{internal}, Bank_{output}, \{Bank_i\}_i \text{ is a } y\text{-input wire})$

Figure 12: Simulator Initialization $SimInit$ **Simulator** $SimEval(state_{t-1}, x_t, C(y, x_t))$

The simulator first computes v'_i values for each wire i in the circuit by evaluating $C(\vec{0}, x_t)$. For each circuit wire i , choose shares (a_i, b_i) for each wire:

x_t input wire corresponding to $x_t[j]$: $a_i \leftarrow x_t[j], b_i \leftarrow 0$

y -input wire: $a_i, b_i \leftarrow 0$

internal wire: $a_i \leftarrow_R \{0, 1\}, b_i \leftarrow v'_i \oplus a_i$

output wire: $a_{output} \leftarrow C(y, x_t), b_{output} \leftarrow v'_{output} \oplus a_{output}$

After the a_i, b_i shares have been computed for each wire, simulate $Eval$ as follows:

- in Step 1, for each wire i , replace each call to $BankGen$ for wire i with a call to $SimBankGen$ with b_i . Replace each call to $BankUpdate$ and $BankRedraw$ with a call to $SimBankUpdate$ or $SimBankRedraw$ (respectively).
- in Step 2, for each NAND gate with input wires i, j and output wire k , compute:

$$a_k \leftarrow SafeNAND(a_i, key_i, \tilde{c}_i^{in}, a_j, key_j, \tilde{c}_j^{in}, key_k, \tilde{c}_k^{out})$$

for each duplication gate with input wire i and output wires j, k , compute:

$$a_j \leftarrow SafeDup(a_i, key_i, \tilde{c}_i^{in}, key_j, \tilde{c}_j^{out})$$

$$a_k \leftarrow SafeDup(a_i, key_i, \tilde{c}_i^{in}, key_k, \tilde{c}_k^{out})$$

- as in $Eval$, the new state is $state_t \leftarrow (Bank_{internal}, Bank_{output}, \{Bank_i\}_i \text{ is a } y\text{-input wire})$

Figure 13: Sim procedure performed on input x_t and circuit output $C(y, x_t)$

joint distributions of the leakage in all of these calls, *together with all keys and ciphertexts produced*, are statistically close in $Real$ and in $HybridReal$. We can complete the generation of the view in both cases (the leakage from $SafeNAND$ and $SafeDup$) as a function of the keys and ciphertexts produced, and we conclude that the two views are statistically close.

***HybridReal* to *HybridReal'* and *Simulated* to *Simulated'*: replacing safe computations with simulated leakage.** Next, we obtain *HybridReal'* from *HybridReal* by replacing the *SafeNAND* calls, in each execution and for each internal and output wire i , with calls to the *SimNAND* simulator from Lemma 6.1, using *HybridReal*'s a_k public shares and the underlying distributions on keys and ciphertexts (the underlying distributions are a function of the leakage values in prior computations).

Similarly, we obtain a hybrid *Simulated'* from *Simulated* by replacing the *SafeNAND* calls with calls to the *SimNAND* simulator, using *SimEval*'s a_i public shares and its underlying distributions on keys and ciphertexts.

Note that, in particular, *HybridReal'* and *Simulated'* can no longer be generated efficiently (because the *SafeNAND* simulator is not efficient). By Lemmas 5.3 and 5.5, the conditions of Lemma 6.1 all hold for each replacement of *SafeNAND* by *SimNAND* in both views (given the leakage in prior computations). In particular, the keys and ciphertexts involved in each *SafeNAND* come from *IuO* distributions whose underlying distributions have high entropy (w.h.p.). This is where we use the fact that, for each internal wire i , even given the leakage, the i -th wire's ciphertexts \bar{c}_i^{out} and \bar{c}_i^{in} are independent up to having the same orthogonality w.r.t. *key*. We conclude that *HybridReal* and *HybridReal'* are statistically close, as are *Simulated* and *Simulated'*.

Closeness of *HybridReal'* and *Simulated'*. *HybridReal'* and *Simulated'* are both obtained as a function of leakage from a sequence of *SimBankGen* calls: the leakage from these generations is then used to compute the leakage for *SimNAND* calls (the leakage from the generations specifies the underlying distributions used by *SimNAND*). In particular, the actual keys and ciphertexts generated are never again accessed after their generation. The same post-processing is performed on the leakage from the generations in both cases: namely, calls to *SimNAND* on the same underlying distributions, and with identically distributed a_i values. The two sequences of generations differ only in the orthogonalities of the underlying plaintexts that are generated in the *SimBankGen* calls for the output and the y -input wires (the plaintexts for internal wires are identically distributed). By Lemma 5.2, we conclude that the leakage from the generations is statistically close in both cases, and so *HybridReal'* and *Simulated'* are also statistically close. ■

References

- [Ajt11] Miklos Ajtai. Secure computation with information leaking to an adversary. In *STOC*, 2011.
- [BCG⁺11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *ASIACRYPT*, pages 722–739, 2011.
- [BGI⁺01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In *CRYPTO*, pages 1–18, 2001.
- [BGJK12] Elette Boyle, Shafi Goldwasser, Abhishek Jain, and Yael Tauman Kalai. Multiparty computation secure against continual leakage. In *STOC*, 2012.

- [BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *CRYPTO*, pages 108–125, 2008.
- [BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.
- [CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.
- [DHLAW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.
- [DLWW11] Yevgeniy Dodis, Allison Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. Cryptology ePrint Archive, Report 2011/369, 2011.
- [DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.
- [DRS04] Yevgeniy Dodis, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT*, pages 523–540, 2004.
- [FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.
- [FRR⁺10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *EUROCRYPT*, pages 135–156, 2010.
- [GIS⁺10] Vipul Goyal, Yuval Ishai, Amit Sahai, Ramarathnam Venkatesan, and Akshay Wadia. Founding cryptography on tamper-proof hardware tokens. In *TCC*, pages 308–326, 2010.
- [GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In *CRYPTO*, pages 39–56, 2008.
- [GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious RAMs. *J. ACM*, 43(3):431–473, 1996.
- [GR10] Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In *CRYPTO*, pages 59–79, 2010.
- [Imp10] Russel Impagliazzo. Personal communication, 2010.
- [ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.
- [JV10] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *CRYPTO*, pages 41–58, 2010.

- [LLW11] Allison Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *STOC*, 2011.
- [LRW11] Allison Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, 2011.
- [MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.
- [NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.
- [Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.
- [Rao07] Anup Rao. An exposition of bourgain’s 2-source extractor. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(034), 2007.
- [Rot12] Guy N. Rothblum. How to compute under ac^0 leakage without secure hardware. In *Manuscript*, 2012.
- [SYY99] Tomas Sander, Adam Young, and Moti Yung. Non-interactive cryptocomputing for nc^1 . In *FOCS*, pages 554–567, 1999.

Multiparty Computation Secure Against Continual Memory Leakage

Elette Boyle
MIT
eboyle@mit.edu

Shafi Goldwasser^{*†}
MIT and Weizmann
shafi@mit.edu

Abhishek Jain
UCLA
abhishek@cs.ucla.edu

Yael Tauman Kalai
Microsoft Research
yael@microsoft.com

ABSTRACT

We construct a multiparty computation (MPC) protocol that is secure even if a malicious adversary, in addition to corrupting $1-\epsilon$ fraction of all parties for an arbitrarily small constant $\epsilon > 0$, can *leak* information about the secret state of each honest party. This leakage can be *continuous* for an unbounded number of executions of the MPC protocol, computing different functions on the same or different set of inputs. We assume a (necessary) “leak-free” preprocessing stage.

We emphasize that we achieve leakage resilience *without weakening the security guarantee* of classical MPC. Namely, an adversary who is given leakage on honest parties’ states, is guaranteed to learn nothing beyond the input and output values of corrupted parties. This is in contrast with previous works on leakage in the multi-party protocol setting, which weaken the security notion, and only guarantee that a protocol which leaks ℓ bits about the parties’ secret states, yields at most ℓ bits of leakage on the parties’ private *inputs*. For some functions, such as voting, such leakage can be detrimental.

Our result relies on standard cryptographic assumptions, and our security parameter is polynomially related to the number of parties.

Categories and Subject Descriptors

C.2.0 [Computer-Communication Networks]: General;
F.0 [Theory]: General

^{*}Supported under NSF Contract CCF-1018064.

[†]This material is based on research sponsored by the Air Force Research Laboratory under agreement number FA8750-11-2-0225. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

STOC’12, May 19–22, 2012, New York, New York, USA.

Copyright 2012 ACM 978-1-4503-1245-5/12/05 ...\$10.00.

Keywords

Cryptography, secure multiparty computation, leakage resilience.

1. INTRODUCTION

The notion of *secure multiparty computation* (MPC), introduced in the works of Yao [Yao82] and Goldreich, Micali and Wigderson [GMW87], is one of the cornerstones in cryptography. Very briefly, an MPC protocol for computing a function f allows a group of parties to jointly evaluate f over their private inputs, with the property that an adversary who corrupts a subset of the parties does not learn anything beyond the inputs of the corrupted parties and the output of the function f . Over the years, MPC protocols have found numerous applications, such as in protocols for auctions, electronic voting, private information retrieval, and threshold and proactive cryptography.

The definition of security for MPC assumes that an adversary sees the messages sent and received by honest parties, but their internal state is perfectly secret. However, over the last two decades, it has become increasingly evident that in the real world, attackers can gain various additional information about the secret states of the honest parties via various *side-channel attacks* (see [Koc96, AK96, QS01, GMO01, OST06, HSH⁺08] and references therein).

In this work, we study MPC in the setting where an adversary, who corrupts an arbitrary subset of parties in the protocol, can also *leak* information about the *entire* secret state of each honest party throughout the protocol execution (except during a designated leak-free preprocessing stage). Leakage is modeled by allowing the adversary to query leakage functions, as follows. Each leakage function is computed by an arbitrary poly-size circuit, with bounded output-length, which is applied to the secret state of an honest processor. The adversary may choose the leakage functions adaptively, based on the entire history of communication, previous leakage, and internal state of corrupted processors.

The security guarantee we aim for and will achieve, is that any adversary in the above leakage model, *does not learn anything beyond the inputs of the corrupted parties and output values of the functions* computed by the MPC protocol. This is formalized via the standard real/ideal world paradigm. In the ideal world, parties do not interact directly, but rather send their inputs to an “ideal functionality”, who computes the function for them, and sends them the output. There is no leakage in the ideal world. An MPC protocol is said to be secure, if for every “real world” leakage adver-

sary \mathcal{A} (as above) there exists an “ideal world” simulator S , such that the output of all the parties (including the adversary) in the real world, is computationally indistinguishable from the output of all the parties (including the simulator) in the ideal world.

Weakly Leakage-Resilient MPC.

We note that recently there have been several results that consider the problem of constructing leakage-resilient protocols [GJS11, BCH11, DHP11, BGG⁺11]. However, in contrast to the security guarantee we consider here, all these results give a *weak* security guarantee (though, they do not rely on a leak-free preprocessing stage). They guarantee that an adversary that runs the protocol and leaks ℓ bits about the honest parties’ secret state, does not learn more than the output of the function being computed, *and an additional ℓ bits* about the private inputs of the honest parties. We note that leakage of ℓ bits on the private inputs of the honest parties could be detrimental to the security of the entire MPC protocol. For example, say the function to be computed by the MPC protocol is to tally up the binary votes of the parties. Then, the ℓ bits can be exactly the complete ℓ votes of any ℓ honest parties, rendering the protocol useless.

Moreover, this weak security notion allows the adversary to learn ℓ bits about the *joint* view of all the honest parties. Thus, another instructive example is to think of the function being computed as a threshold decryption function, where each party has a secret-share of the decryption key. In this case, the weak security guarantee allows the leakage of ℓ bits from the decryption key, which for some decryption algorithms could entirely compromise security.

Interestingly, we use the result in [BGG⁺11], which constructs an MPC protocol with the weak security guarantee, as a building block to construct a leakage-resilient MPC protocol with the classical (strong) security guarantee.

Security Against Continual Leakage.

We further remark that the weaker security notion previously achieved cannot be extended meaningfully to continual leakage in the MPC setting. That is, it cannot address the setting where the n users do not just perform a one-shot MPC protocol, but rather engage in an unbounded number of MPC protocols for many functions, and during each MPC invocation the adversary leaks ℓ bits from each of the honest party’s internal state. This is obvious, as allowing the repeated leakage of new ℓ bits of information on the honest parties’ inputs would eventually leak the honest parties’ inputs in their entirety. For example, in the setting where a set of parties jointly compute a threshold decryption function (as described above), they may want to carry out many decryption computations, where leakage happens repeatedly. Since each ℓ bits of leakage corresponds to ℓ bits of leakage on the decryption key, the decryption key may eventually be completely leaked! Nonetheless, we use the result of [BGG⁺11] as a building block to achieve our stronger *continual leakage* security guarantee.

1.1 Our Result: Continual Leakage-Resilient MPC

In this work, we construct a leakage-resilient MPC protocol for any function f , *without weakening the security guarantee*. We consider a continual setting, where parties over time compute many functions on their inputs. Our security

guarantee is that an adversary does not learn *anything* beyond the inputs of the corrupted parties and the output of the functions computed, even if he continually leaks information about the honest parties’ secret states throughout the protocol executions. Parties’ secret states are periodically updated via an update procedure, during which the adversary can continue to leak information. We allow each of the adversary’s leakage functions to be an *arbitrary* (shrinking) polynomial time computable function of the *entire* secret state of each honest party (separately), and these leakage functions can be chosen adaptively on all information the adversary has seen thus far.

Theorem (Informal).

Under (standard) intractability assumptions, for every constant $\epsilon > 0$ there exists an MPC protocol for computing an unbounded number of functions among n parties of which at least ϵ fraction are honest. The protocol is secure against *continual* leakage, assuming a *one-time* leak-free preprocessing stage in which the inputs are shared, and where the security parameter is polynomially related to the number of parties n .

A few remarks about our result statement are in order.

“Leak-free” Preprocessing.

We assume the existence of a leak-free preprocessing stage. We stress that this is a *necessary* assumption to obtain our strong security guarantee, since otherwise an adversary can simply leak ℓ bits about an honest party’s secret input, before the MPC even commences. More generally, we note that such a leak-free preprocessing stage is a necessary step in the construction of any leakage resilient cryptographic primitive which receives a secret input, and where the security guarantee is that the secret input does not leak. This is the case, for example, in the compilers of [ISW03, FRR⁺10, JV10, GR10, GR12], which transform algorithms with a secret state into a functionally equivalent leakage-resilient variant of the same algorithm.

We remark that our preprocessing stage in fact has the nice property that it can be decomposed into two parts, namely, (a) an interactive preprocessing phase that is *independent* of the parties’ inputs and the functions to be computed, and (b) a *non-interactive* input dispersal phase. We stress that the first phase is run only *once* in the beginning of time, before the parties know what their inputs are or what functions they wish to compute. The second (non-interactive) phase is run whenever the parties choose a set of inputs.

While both of these parts are assumed to be “leak-free”, we do allow leakage between them. We refer the reader to Section 3 for a formal description of our model.

Multi-function MPC and Continual Leakage.

We note that in the standard (leak-free) MPC literature, one typically considers a one-shot MPC protocol, as opposed to considering the setting where the parties compute an unbounded (polynomial) number of functions. The reason we focus on the latter setting, is to emphasize that we need to run the leak-free preprocessing stage only *once*, and then the parties can compute any unbounded number of functions f_1, \dots, f_ℓ in a leaky environment.

We further emphasize that we allow the adversary to leak

continuously on the secret states of the parties during the unbounded computations; the only (necessary) requirement is that the secret states of the parties are periodically updated (since otherwise they will eventually be completely leaked). However, the adversary is allowed to leak even *during* each update procedure. We do not bound the total number of bits that the adversary leaks, but rather only bound the leakage rate: i.e., the number of bits leaked *between updates*.

Extending to Multi-Input MPC.

We stated our theorem for the case of computing many functions on a single set of inputs. However, our construction is easily extended to the many-input case. Whenever a party chooses a new input, the (leak-free) non-interactive input phase described above can be repeated. Namely, party P_i on new input x_i performs a local computation on x_i , sends a message to the other parties, and erases x_i . One may think of this model as a “hot potato” model, where the parties never store their inputs for very long (since they are concerned with leakage), but rather immediately share their input (as if it were a “hot potato”).

Number of Parties vs. Security Parameter.

Notice that in our theorem, the security parameter is polynomially related to the number of parties. Namely, the security increases with the number of parties. Therefore, this theorem is meaningful only when the number of parties in the MPC protocol is large. One may ask whether this restriction on the number of parties being large, or the restriction that an ϵ -fraction is honest, is inherent, or whether it is simply an artifact of our techniques. Unfortunately, it turns out that this restriction cannot be removed altogether. In particular, one can prove that there does not exist a secure leakage-resilient *two-party* computation protocol in our model.¹ Similarly, one can show that there does not exist a secure leakage-resilient MPC protocol if all the parties except one are malicious. Moreover, jumping ahead, in Section 1.4 we show that proving this theorem for constant number of parties, implies an “only computation leaks (OCL) compiler” (without leak-free hardware) that has only a constant number of sub-computations (or “modules”), which is an interesting open problem on its own. We refer the reader to Section 1.4 for details.

Assumptions.

In our construction, we rely on several underlying cryptographic primitives, including a fully homomorphic encryption (FHE) scheme [Gen09, BGV11], a non-interactive zero-knowledge (NIZK) proof-of-knowledge system [FLS90], a standard MPC protocol [GMW87], an equivocal commitment scheme [FS89], a weakly leakage-resilient MPC protocol [BGG⁺11],

and an LDS compiler [BCG⁺11] (which can be thought of as a stronger version of an OCL compiler as in [JV10, GR10, GR12]). These primitives have been shown to exist under various standard computational intractability assumptions, and we refer the reader to Section 2 for details on these primitives, and the corresponding assumptions. We note however that all these primitives, excluding FHE, can be based on the DDH assumption.

The use of FHE in our construction is in order to ensure the number of parties required will be independent of the complexity of the functions computed by the MPC protocol.²

Applications.

We demonstrate the application of our result to the problem of delegating multi-party computation to outside servers. Generally, the setting is of a large set of parties who need to perform a joint computation, and they would like a service (such as Amazon) to do the computation for them. However, they do not trust any one server, and further believe that any server can be leaked upon.

Usually, MPC provides a solution around the trust problem by using several servers, as follows: Each party secret shares her input, and gives one share to each server; then the servers carry out the desired computation by running an MPC protocol; finally, one argues that if there are sufficiently many honest parties, then security is guaranteed. However, if an adversary can obtain leakage information from the honest servers, then this is no longer true. To argue security in the leaky setting, the servers will need to run a *leakage-resilient* MPC protocol. Moreover, if the servers compute many functions on the secret inputs, then they will need to run an MPC protocol that is secure against *continual leakage*. Let us demonstrate three examples of this setting.

- *Electronic election*: Say an electronic election among many voters is to be held. Clearly running an MPC protocol among all voters is prohibitive, since it requires interaction between every two voters. Instead, the MPC protocol is run by a proxy of n servers. Since these servers compute on very sensitive information, attackers may try to employ various side-channel attacks to learn this information. Thus, to ensure the secrecy of the individual votes, the servers should run a *leakage-resilient* MPC protocol.
- *Medical Data*: One may envision a huge database which contains the medical data of every patient in the US. To compute any global statistic on this data, one would not want to put complete trust in any single database.

¹The reason is the following: Assume the adversary controls party P_1 . In this case, he knows the entire secret state s_1 of P_1 , and can choose his leakage function L to depend on s_1 : i.e., $L = L_{s_1}$. Note that L takes as input the secret state s_2 of P_2 , and thus the adversary can leak any (shrinking) function $g(s_1, s_2)$ by setting $L_{s_1}(s_2) \triangleq g(s_1, s_2)$. But, recall that from the secret states (s_1, s_2) the parties can compute any function of the original inputs (x_1, x_2) . Therefore, the function leaked can be an arbitrary function of the original inputs. Clearly, such leakage cannot be simulated in the ideal world.

²We emphasize that, while FHE immediately solves the related problem of computing on encrypted data, FHE does *not* suffice for our purposes. To illustrate, suppose the parties collectively generate a public key pk for the FHE scheme, so that they each hold a secret share of the corresponding secret key, and then each publish an encryption of their input x_i . Then for any efficiently computable function f , they can easily produce an *encryption* of the desired output, $\text{Enc}_{\text{pk}}(f(\vec{x}))$. However, the challenge is (even for a one-shot function computation) how to enable the parties to collectively *decrypt* this ciphertext and reveal $f(\vec{x})$ itself, while simultaneously ensuring that the adversary (who can corrupt nearly all of the parties, and leak on all the rest) is not able to learn any information on the x_i 's.

Instead, it is distributed to n different databases. Each time they need to compute statistics on this data, they engage in an MPC protocol. As in the voting example, since these databases contain very sensitive information, an adversary may try to obtain this information via a leakage attack. Thus, to ensure security, the databases must run an MPC protocol that is secure against *continual leakage*.

- *Differential Privacy*: In the area of differential privacy, great care is taken to ensure that the data of individuals is protected. However, usually it is assumed that there is an *honest* curator, and that the people in the database hand their secret data to this curator. However, it seems likely that people may not trust any single curator with highly sensitive information (such as whether they do or do not have a disease which may scare off life insurance providers). Thus, as in the previous examples, this trusted curator can be replaced by a multitude of parties of which only a small fraction is assumed to be honest. Moreover, if these parties compute on the database using a leakage-resilient MPC protocol, then security is guaranteed even if all the honest parties are leaked upon (as long as some ϵ -fraction of the honest parties are not *fully* leaked upon).

1.2 Related Work

Leakage-Resilient Non-Interactive Primitives.

There has been an extensive amount of research on leakage-resilient cryptography in the past few years. Most prior works construct specific leakage-resilient *non-interactive* primitives, such as leakage-resilient encryption schemes and leakage-resilient signature schemes [DP08, AGV09, Pie09, DKL09, ADW09, NS09, KV09, DGK⁺10, FKPR10, ADN⁺10, KP10, GR10, JV10, BG10, BKKV10, DP10, DHLW10a, DHLW10b, LRW11, MTVY11, BSW11, LLW11, DLWW11, BCG⁺11].

Weakly Leakage-Resilient Interactive Protocols.

There has also been prior work on the problem of constructing leakage-resilient interactive protocols [GJS11, BCH11, BGK11, DHP11, BGG⁺11]. Garg *et al.* [GJS11] present a leakage-resilient zero-knowledge proof system. Bitansky *et al.* [BCH11] present leakage-resilient protocols for various functionalities (such as secure message transmission, oblivious transfer, and commitments) which are secure against semi-honest adversaries, and also zero knowledge, in the UC framework. Boyle *et al.* [BGK11] present a leakage-resilient multi-party coin tossing protocol. Dămgard, Hazay, and Patra [DHP11] present a general leakage-resilient two-party secure function evaluation protocol for NC^1 functions in the *semi-honest* setting. In their model, they further place a restriction that the adversary must leak on the input and randomness of an honest party's secret state *independently*. Finally, very recently Boyle *et al.* [BGG⁺11] constructed a general leakage-resilient MPC protocol that is secure in the UC setting.

However, all the results in the interactive setting mentioned above offer a *weak* security guarantee, that an adversary that leaks ℓ bits in the real world, gains at most ℓ bits of secret information about the secret inputs of the parties. (An exception is the work of [BGK11] that considered the

specific coin-tossing functionality, where the parties do not have any secret inputs.) Moreover, the ℓ bits of secret information gained is an arbitrary (poly-size) function of the *joint* inputs x_1, \dots, x_n .

Only Computation Leaks Model.

Finally, we mention that various leakage models have been considered in the literature that restrict the leakage functions in different ways. Most notable is the *only computation leaks* (OCL) model of Micali and Reyzin [MR04]. The axiom of this model is that secret information that is merely stored in memory does not leak, but any information that is used during a computation may leak.

Several results prove security for specific cryptographic primitives in the OCL leakage model [DP08, Pie09, FKPR10]. More generally, it is known how to convert *any* circuit into one that is secure in the OCL model [GR10, JV10, GR12]. In particular, a recent work of Goldwasser and Rothblum [GR12] shows how to do this unconditionally, making no intractability assumptions, and without resorting to secure leak-free hardware, unlike the previous works. Specifically, Goldwasser and Rothblum construct an efficient compiler that takes any circuit (with some secret values hard-wired) and converts it into a leakage-resilient one, consisting of several *modules*, each of which performs a specific sub-computation. The security guarantee is that an adversary, who at any point of time throughout the computation obtains bounded leakage from the “currently active” module, does not learn any more information than having black-box access to the circuit. We will use a variant of this result (namely, an LDS compiler; see Section 2.5) to construct our leakage-resilient MPC scheme. In particular, we use [GR12] as a building block in our construction. See Section 1.3 for details.

We stress that our result does *not* use the OCL assumption, and we allow the adversary to compute leakage functions on *everything* held in the memory of each party (except during the preprocessing phase and during the input phase).

1.3 Overview of Our Construction

Starting point – OCL Compiler.

As discussed earlier, it is known how to convert *any* circuit into one that is secure in the *only computation leaks* (OCL) model (without assuming secure hardware) [GR12]. In light of this result, a natural first idea toward realizing our goal of constructing leakage-resilient MPC protocols, is the following. Let P_1, \dots, P_n denote the set of all parties, and let $U_{\vec{x}}$ be a universal circuit that has the secret input vector \vec{x} of all the parties hard-wired into it and on input a circuit f outputs $U_{\vec{x}}(f) = f(x)$. Then, very roughly, the candidate MPC protocol works as follows. First, in the “leak free” preprocessing phase, apply the OCL compiler of [GR12] on circuit $U_{\vec{x}}$ to obtain a set of modules $\text{Sub}_1, \dots, \text{Sub}_n$ such that on any input f , the “compiled” circuit (consisting of $\text{Sub}_1, \dots, \text{Sub}_n$) outputs $U_{\vec{x}}(f) = f(\vec{x})$. Next, in the computation phase, in order to securely compute a function f , each party P_i *emulates* the module Sub_i (such that the computation of Sub_i is performed by party P_i), where the input of Sub_1 is f , and the output of Sub_n is the protocol output $f(\vec{x})$. Finally, in the update phase, the parties update their respective modules by running the update algorithm of the OCL compiler.

Now, assuming that we can reduce (independent) leakage

on each party to (independent) leakage on its corresponding module, one may hope that the above MPC protocol achieves the desired security properties: in particular, privacy of the inputs that were “encoded” in the preprocessing phase. Unfortunately, as we explain below, this is not the case. Nevertheless, as will be evident from the forthcoming discussion, the above approach serves as a good starting point toward realizing our goal.

OCL Compiler vs. LR-MPC.

There are two main differences between the setting of leakage-resilient MPC (LR-MPC) and an OCL compiler.

1. The first difference is perhaps best illustrated by the fact that an OCL compiler only guarantees security against an *external* adversary who can obtain leakage from the modules. In contrast, in the setting of LR-MPC, we wish to guarantee security against an *internal* adversary, who may also corrupt a subset of the parties.

More concretely, recall that the security of the OCL compiler crucially relies on the assumption that an external adversary can only obtain *bounded, independent* leakage on each module. Further, in order for the correctness of the compiled circuit to hold, each module must perform its computation as specified. As a result, the above approach, at best, yields an MPC protocol that is secure when *all* the parties are honest (not even semi-honest) but can be leaked upon by an external adversary. Specifically, note that if an internal adversary can corrupt some of the parties, then we can no longer guarantee correctness of computation, and even worse, an adversary may be able to obtain *joint* leakage on multiple modules, and learn the *entire* secret state of modules corresponding to corrupted parties, thus violating both of the above stated requirements.

2. The second difference between the OCL compiler and the leakage-resilient MPC setting is that in the OCL setting, the communication between the modules is assumed to be private (but may be leaked), and leakage is assumed to happen “in order”; i.e., only a module which is currently computing can be leaked upon. On the other hand, in the leakage-resilient MPC setting, the entire communication is to be known to the adversary, and moreover, leakage on any party can happen at any time.

Emulating Modules via Weakly LR-MPC.

Our key idea to circumvent the first problem stated above is to emulate each Sub_i by a designated *set* of parties $S_i = \{P_{i_1}, \dots, P_{i_\ell}\}$, instead of a single party P_i . More concretely, we secret share Sub_i between $P_{i_1}, \dots, P_{i_\ell}$, who then run a specific MPC protocol Π to jointly emulate the (functionality of) module Sub_i . Now, note that as long as at least one of the parties in the designated set S_i is honest, the emulation of Sub_i will be “correct”, and if leakage on each honest party is bounded, then we can expect the leakage on the module Sub_i to be bounded as well. Furthermore, if all of the designated sets S_i for the modules Sub_i are disjoint (i.e., no party is contained within two different sets), then the leakage on each module will be independent, as required. However, note that since we are in the setting of leakage, in

order for the above idea to work, we need the MPC protocol Π to satisfy some form of leakage-resilience. Thus, a priori, it seems that we haven’t made any progress at all.

Our next crucial observation is that protocol Π in fact only needs to satisfy a *weaker* form of leakage-resilience. Specifically, we only require that leakage on the secret state of each party P_{i_ℓ} executing protocol Π (to emulate Sub_i) can be “*reduced*” to leakage on the module Sub_i . (This suffices since the OCL compiler allows bounded leakage on each module.) More generally, this translates to constructing an MPC protocol such that the leakage on the secret states of the honest parties in the real world can be reduced to leakage on the inputs of the honest parties in the ideal world. Fortunately, an MPC protocol (for any poly-size function f) satisfying the above (weak) form of leakage-resilience was recently constructed by Boyle *et al.* [BGG⁺11]. Thus, we are able to employ their construction here.³

However, the result of Boyle *et al.* is only for *deterministic* functions, whereas the modules in the OCL construction compute *randomized* functions. Thus, we need to extend the weakly leakage-resilient MPC to hold for randomized computations. See Section 2.6 (and Section 2.6.1 in particular) for further details.

Using an LDS Compiler Instead of an OCL Compiler.

Our key idea to circumvent the second problem stated above is to use an LDS compiler instead of an OCL compiler. The LDS (leaky distributed system) model was introduced in [BCG⁺11], and it strengthens the OCL model in two ways (which are exactly the strengthenings we need). First, in the OCL model, leakage occurs in a certain ordering (based on the order of computation). The LDS model strengthens the power of the adversary, by allowing him to leak from the sub-computations in any order he wishes. Moreover, he can leak a bit from Sub_i , then leak a bit from Sub_j , and based on the leakage values, leak again on Sub_i . So, the adversary controls which Sub_i he wishes to leak from. In addition, in the LDS model, the adversary can view and control the entire communication between the modules. We refer the reader to Section 2.5 for details on the LDS compiler.

By using an LDS compiler, as opposed to an OCL compiler, we get around the second problem mentioned above.

Reducing Number of Parties via FHE.

An important issue that was overlooked in the previous discussion is the following. The only known OCL compiler that does not rely on leak-free hardware [GR12], and thus the only known LDS compiler without leak-free hardware, suffers from the drawback that the number of modules in the “compiled” circuit is linear in the size of the original circuit. As a result, when we apply the LDS compiler on $U_{\vec{x}}$, whose size grows with $|\vec{x}|$, the number of resultant modules is more than the number of parties! Thus, a priori, it is not even clear how to realize the above approach.

³At this point, an advanced reader may question whether the result of Boyle *et al.* [BGG⁺11], in conjunction with a leakage-resilient secret sharing scheme, *directly* yields a leakage-resilient MPC protocol in our model. Unfortunately, this is not the case since the simulator of Boyle *et al.* requires *joint* leakage on the honest party inputs, even when the real world adversary makes *disjoint* leakage queries on the secret states of honest parties. We refer the reader to Section 1.4 for more details.

In order to resolve this above problem, we make crucial use of fully homomorphic encryption (FHE) in the following manner. Instead of simply applying the LDS compiler to $U_{\hat{x}}$, we now first compute a key pair (pk, sk) for an FHE scheme, and then apply the LDS compiler to the decryption circuit $\text{Dec}_{sk}(\cdot)$ with the secret key sk hardwired. Note that the number of resultant modules is now *independent* of the number of parties. Now, in a non-interactive input phase (that is also “leak-free”), the parties P_i each encrypt their respective inputs x_i under the public key pk , and publish the resulting ciphertexts \hat{x}_i . Then, whenever the parties wish to compute a functionality f over their inputs, they homomorphically evaluate $\hat{y}_f \triangleq \text{Eval}_{pk}((\hat{x}_1, \dots, \hat{x}_n), f)$, and collectively evaluate the compiled decryption circuit on the value \hat{y}_f in the manner described above.

We note that the use of FHE allows us to obtain the desired property that the preprocessing phase is *independent* of the inputs and functions to be computed, since in this phase a key pair (pk, sk) is generated and the LDS compiler is applied to the corresponding decryption circuit $\text{Dec}_{sk}(\cdot)$. In addition, the input phase is non-interactive, since in this phase the parties simply compute and send an encryption of their inputs.

Missing Pieces.

A few technical issues still remain undiscussed. For example, it is not immediately clear how to choose the designated sets of parties S_i such that at least one of the parties in each set S_i is honest, and each set S_i is independent. Very roughly, to address this problem, we employ (an adapted version of) the committee election protocol of Feige [Fei99] to divide the parties into several committees, one for each module. Then, by a careful choice of parameters, we are able to obtain the desired guarantees. We refer the reader to the technical sections for more details.

1.4 Future Directions

LR-MPC for Constant Number of Parties.

Perhaps the most interesting open question left from this work is to construct a leakage-resilient MPC protocol for *constant* number of parties. We note that such a result (even if we only consider adversaries that leak, but do not corrupt any party) will imply the following interesting corollary: The existence of an efficient compiler that converts any circuit into a leakage-resilient circuit that is secure in the “only computation leaks” (OCL) model with *constant* number of modules (and without assuming leak-free hardware). We refer the reader to Section 2.5 for details.

To see this implication, consider such a leakage-resilient MPC protocol. Let (an arbitrary) party P_1 take as his secret input the secret circuit C to be compiled, and the other parties take no inputs. After the leak-free preprocessing stage (and the leak-free input stage), each party P_i holds a secret state s_i . We think of each party P_i as being a module Sub_i in the compiled circuit. To evaluate the circuit C on (public) input x , the modules carry out a leakage-resilient MPC computation of the universal function U_x , that on inputs $\{s_i\}$, which form some sort of secret-sharing of C , outputs $C(x)$. Since the OCL model allows leakage on each module separately, this corresponds to allowing leakage on each party separately, which according to our definition of

security gives no information about the secret C beyond the output value $C(x)$.

Weakly Leakage-Resilient MPC with Disjoint Leakage.

Another interesting open question is to construct a leakage-resilient MPC protocol without assuming any leak-free stages, and requiring the following weakened security definition: For each “real world” adversary that makes ℓ leakage queries, where each leakage query is applied to the secret state of a *single* honest party, there exists a simulator in the “ideal world” that makes at most ℓ leakage queries, where each leakage query is applied to the input of a *single* honest party.

We note that the recent result of [BGG⁺11] allowed the adversary in the “real world” to make leakage queries on the *joint* secret state of all the parties, and allowed the simulator in the “ideal world” to make leakage queries that are a function of all the inputs of the honest parties. Unfortunately, their simulator requires joint leakage on the honest party’s inputs even in the case where the adversary only makes disjoint leakage queries.

We next show that such a leakage-resilient MPC protocol, where the leakage in the real world and in the ideal world is made on each party separately, would imply a result similar to ours, which allows a leak-free preprocessing stage, but considers a strong security guarantee. Intuitively, in the leak-free preprocessing stage, the parties will secret share their inputs via a secret sharing scheme that is resilient to continual leakage. Such a scheme was recently presented by Dodis *et al.* [DLWW11]. Then, any time the parties wish to compute a function f of their secret inputs, they will run the weak leakage-resilient MPC protocol. Security follows from the fact that the adversary only gains leakage from the secret share of each party *separately*, and from the fact that the secret-sharing scheme is resilient to continual leakage on each of its shares.

LR-MPC with Non-Interactive Preprocessing.

Finally, an interesting open question that is left by this work, is to construct a leakage-resilient MPC protocol without the initial leak-free preprocessing stage, but only with the leak-free *non-interactive* input stages.

2. PRELIMINARIES

2.1 Non-Interactive Zero Knowledge

DEFINITION 2.1. [FLS90, BFM88, BSMP91]:

$\Pi = (\text{Gen}, \text{P}, \text{V}, \mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{proof}}))$ is an efficient adaptive NIZK proof system for a language $L \in \text{NP}$ with witness relation \mathcal{R} if $\text{Gen}, \text{P}, \text{V}, \mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{proof}}$ are all ppt algorithms, and there exists a negligible function μ such that for all k the following three requirements hold.

- **Completeness:** For all x, w such that $\mathcal{R}(x, w) = 1$, and for all strings $\text{crs} \leftarrow \text{Gen}(1^k)$,

$$\text{V}(\text{crs}, x, \text{P}(x, w, \text{crs})) = 1.$$

- **Adaptive Soundness:** For all adversaries \mathcal{A} , if $\text{crs} \leftarrow \text{Gen}(1^k)$ is sampled uniformly at random, then the probability that $\mathcal{A}(\text{crs})$ will output a pair (x, π) such that $x \notin L$ and yet $\text{V}(\text{crs}, x, \pi) = 1$, is at most $\mu(k)$.

- **Adaptive Zero-Knowledge:** For all ppt adversaries \mathcal{A} ,

$$|\Pr[\text{Exp}_{\mathcal{A}}(k) = 1] - \Pr[\text{Exp}_{\mathcal{A}}^S(k) = 1]| \leq \mu(k),$$

where the experiment $\text{Exp}_{\mathcal{A}}(k)$ is defined by:

$$\begin{aligned} \text{crs} &\leftarrow \text{Gen}(1^k) \\ \text{Return } \mathcal{A}^{\text{P}(\text{crs}, \cdot, \cdot)}(\text{crs}) \end{aligned}$$

and the experiment $\text{Exp}_{\mathcal{A}}^S(k)$ is defined by:

$$\begin{aligned} (\text{crs}, \text{trap}) &\leftarrow \mathcal{S}^{\text{crs}}(1^k) \\ \text{Return } \mathcal{A}^{S'(\text{crs}, \text{trap}, \cdot, \cdot)}(\text{crs}), \end{aligned}$$

where $S'(\text{crs}, \text{trap}, x, w) = \mathcal{S}^{\text{proof}}(\text{crs}, \text{trap}, x)$.

We next define the notion of a NIZK proof of knowledge.

DEFINITION 2.2. Let $\Pi = (\text{Gen}, \text{P}, \text{V}, \mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{proof}}))$ be an efficient adaptive NIZK proof system for an NP language $L \in \text{NP}$ with a corresponding NP relation \mathcal{R} . We say that Π is a proof-of-knowledge if there exists a ppt algorithm E such that for every ppt adversary \mathcal{A} ,

$$\begin{aligned} \Pr[\mathcal{A}(\text{crs}) = (x, \pi) \text{ and } E(\text{crs}, \text{trap}, x, \pi) = w^* \\ \text{s.t. } \mathbf{V}(\text{crs}, x, \pi) = 1 \text{ and } (x, w^*) \notin \mathcal{R}] = \text{negl}(k), \end{aligned}$$

where the probabilities are over $(\text{crs}, \text{trap}) \leftarrow \mathcal{S}^{\text{crs}}(1^k)$, and over the random coin tosses of the extractor algorithm E .

LEMMA 2.3 ([FLS90]). Assuming the existence of enhanced trapdoor permutations, there exists an efficient adaptive NIZK proof of knowledge for all languages in NP.

2.2 Equivocal Commitments

Informally speaking, a bit-commitment scheme is *equivocal* if it satisfies the following additional requirement. There exists an efficient simulator that outputs a fake commitment such that: (a) the commitment can be decommitted to both 0 and 1, and (b) the simulated commitment and decommitment pair is indistinguishable from a real pair. We now formally define the equivocability property for bit-commitment schemes in the CRS model.

The following definition is adapted from [FS89, CIO98].

DEFINITION 2.4. A non-interactive bit-commitment scheme $(\text{Gen}, \text{Com}, \text{Rec})$ in the CRS model is said to be an equivocal bit-commitment scheme in the CRS model if there exists a PPT simulator algorithm $\mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{com}})$ such that \mathcal{S}^{crs} takes as input the security parameter 1^k and outputs a CRS and trapdoor pair, $(\text{crs}, \text{trap})$; and \mathcal{S}^{com} takes as input such a pair $(\text{crs}, \text{trap})$ and generates a tuple (c, d^0, d^1) of a commitment string c and two decommitments d^0 and d^1 (for 0 and 1), such that the following holds.

1. For every $b \in \{0, 1\}$ and every $(c, d^0, d^1) \leftarrow \mathcal{S}^{\text{com}}(\text{crs}, \text{trap})$, it holds that

$$\text{Rec}(\text{crs}, c, d^b) = b.$$

2. For every $b \in \{0, 1\}$, the random variables

$$\{(\text{crs}, c, d) : \text{crs} \leftarrow \text{Gen}(1^k), (c, d) \leftarrow \text{Com}(\text{crs}, b)\}$$

and

$$\begin{aligned} \{(\text{crs}, c, d^b) : (\text{crs}, \text{trap}) \leftarrow \mathcal{S}^{\text{crs}}(1^k), \\ (c, d^0, d^1) \leftarrow \mathcal{S}^{\text{com}}(\text{crs}, \text{trap})\} \end{aligned}$$

are computationally indistinguishable.

Reusable CRS.

Note that the simulator algorithms \mathcal{S}^{crs} and \mathcal{S}^{com} are described as separate algorithms in the Definition 2.4 to highlight that it is not necessary to create a separate CRS for every equivocal commitment, i.e., the CRS is *reusable*. In this case, Definition 2.4 can be extended in a straightforward manner to consider indistinguishability of an honestly generated tuple consisting of a crs and polynomially many commitment-decommitment pairs, from a simulated tuple.

LEMMA 2.5 ([CLOS02]). Assuming the existence of one-way functions, there exists an equivocal bit commitment in the (reusable) CRS model.

String Equivocal Commitments.

For our purposes, we actually use *string* equivocal commitment schemes. Note that such a scheme can be easily constructed by simply repeating the above bit commitment scheme in *parallel*. More specifically, a commitment to a string of length n is a vector (c_1, \dots, c_n) , with corresponding decommitment vector (d_1, \dots, d_n) . The simulator algorithm \mathcal{S}^{com} produces a commitment vector and a *pair* of decommitment vectors $d^0 = (d_1^0, \dots, d_n^0)$, $d^1 = (d_1^1, \dots, d_n^1)$. A decommitment to any particular bit string $a = (a_1, \dots, a_n)$ can be formed by selecting the appropriate decommitment values $(d_1^{a_1}, \dots, d_n^{a_n})$. We denote this vector as d^a .

2.3 The Elect Protocol

As part of our protocol, we elect disjoint committees, and need the guarantee that (with overwhelming probability in k) the number of parties in each committee is of the correct approximate size, and that a constant fraction of each committee is honest. Such a protocol can be obtained using the technique of Feige's lightest bin committee election protocol [Fei99].

Feige's protocol selects a single committee of approximate size k out of n parties by having each party choose and broadcast a random bin in $[\frac{n}{k}]$.⁴ The elected committee \mathcal{E} consists of the parties in the lightest bin. Feige demonstrated that no set of malicious parties $M \subset [n]$ of size $(1 - \epsilon)n$ can force a committee \mathcal{E} to be elected for which $|\mathcal{E} \cap M|$ is significantly greater than $(1 - \epsilon)k$, by using a Chernoff bound to argue that each bin contains nearly ϵk honest parties.

Suppose we wish to elect m disjoint committees, each of size approximately k , where k is the security parameter, and where the number of parties n is at least $n \geq mk^2$. We consider the following protocol, **Elect**. Each party samples a random value $x_i \leftarrow [\frac{n}{k}]$. The resulting committees are precisely the m lightest bins. Namely, suppose the lightest bin is ℓ_1 , the second lightest bin is ℓ_2 , etc. Then $\mathcal{E}_j = \{P_i : x_i = \ell_j\}$, for $j = 1, \dots, m$.

⁴In Feige's original work [Fei99], he considered the specific case of $k = \log n$. For our purpose, we need to elect committees whose size depends on the security parameter (to achieve negligible error), and thus we consider general k .

LEMMA 2.6. Let $n \geq mk^2$, and let $M \subset [n]$ be any subset of corrupted parties of size $(1 - \epsilon)n$. Then the protocol **Elect** yields a collection of m disjoint committees $\{\mathcal{E}_j\}_{j=1}^m$ such that the following properties simultaneously hold with probability $\geq 1 - e^{-\Theta(k)}$:

1. $\forall j, \frac{\epsilon}{2}k \leq |\mathcal{E}_j| \leq (1 + o(1))k$,
2. $\forall j, \frac{|\mathcal{E}_j \cap M|}{|\mathcal{E}_j|} < 1 - \frac{\epsilon}{3}$.

The proof of Lemma 2.6 is very similar to that of the disjoint committee election protocol of [BGK11]. We refer the reader to the full version for a complete analysis.

We remark that a constant fraction of honest parties in the elected committees will be needed for the weakly leakage-resilient MPC for *randomized functionalities* (see discussion in Section 2.6.1).

2.4 Fully Homomorphic Encryption

A fully homomorphic public-key encryption scheme (FHE) consists of algorithms (**Gen**, **Enc**, **Dec**, **Eval**). The first three are the standard key generation, encryption and decryption algorithms of a public key scheme. The additional algorithm **Eval** is a deterministic polynomial-time algorithm that takes as input a public key pk , a ciphertext $\hat{x} \leftarrow \text{Enc}_{\text{pk}}(x)$ and a circuit C , and outputs a new ciphertext $c = \text{Eval}_{\text{pk}}(\hat{x}, C)$ such that $\text{Dec}_{\text{sk}}(c) = C(x)$, where sk is the secret key corresponding to the public key pk . It is required that the size of c depends polynomially on the security parameter and the length of the output $C(x)$, but is otherwise independent of the size of the circuit C .

Several such FHE schemes have been constructed, starting with the seminal work of Gentry [Gen09]. Recently, new schemes were presented by Brakerski, Gentry and Vaikuntanathan [BV11, BGV11] that achieve greater efficiency and are based on the LWE assumption. We note that in these schemes, the size of the public key depends linearly on the depth of the functions being evaluated. As a result, the complexity of our preprocessing phase depends on the maximum depth of functions that we would like to compute. This issue can be avoided altogether if we assume that the schemes of [BV11, BGV11] are circular secure.

For our construction, we need an FHE scheme with the following additional property, which we refer to as *certifiability*. Loosely speaking, an FHE scheme is said to be certifiable, if there is an efficient algorithm that takes as input a random string r and tests whether it is “good” to use r as randomness in the encryption algorithm **Enc**. More precisely, a certifiable FHE scheme is associated with a set R , which consists of all the “good” random strings, such that (1) a random string is in R with overwhelming probability; and (2) The **Eval** algorithm and the decryption algorithm **Dec** are correct on ciphertexts that use randomness from R to encrypt. A formal definition follows.

DEFINITION 2.7. A FHE scheme is said to be certifiable if there exists a subset $R \subseteq \{0, 1\}^{\text{poly}(k)}$ of possible randomness values for which the following hold.

1. $\Pr[r \in R] = 1 - \text{negl}(k)$, where the probability is over uniformly sampled $r \leftarrow \{0, 1\}^{\text{poly}(k)}$.
2. There exists an efficient algorithm \mathcal{A}_R such that $\mathcal{A}_R(r) = 1$ for $r \in R$ and 0 otherwise.

3. We have

$$\Pr_{\text{pk}, \text{sk}} \left[\begin{array}{l} \forall b_1, \dots, b_n \in \{0, 1\}, \forall r_1, \dots, r_n \in R, \\ \forall \text{ poly-size circuits } f : \{0, 1\}^n \rightarrow \{0, 1\} \\ \text{Dec}_{\text{sk}}(\text{Eval}_{\text{pk}}(f, c_1, \dots, c_n)) = f(b_1, \dots, b_n), \\ \text{where } c_i = \text{Enc}_{\text{pk}}(b_i; r_i) \end{array} \right] = 1 - \text{negl}(k).$$

We note that this property holds, for example, for the schemes of [BV11, BGV11]. For the readers who are familiar with these constructions, the set of “good” randomness R corresponds to encrypting with sufficiently “small noise.”

2.5 Leaky Distributed Systems

One of the tools in our construction is a compiler that converts any circuit C (with secrets) into a collection of sub-computations (or “modules”) $\text{Sub}_1, \dots, \text{Sub}_m$, whose sequential evaluation evaluates the circuit C , and which is secure in the *leaky distributed systems* (LDS) model, a model recently introduced by Bitansky *et. al.* [BCG⁺11].

Before we describe this model and compiler, let us recall prerequisite prior works [JV10, GR10, GR12], which construct such a compiler in the “*only computation leaks*” (OCL) model. In particular, these works demonstrate a compiler that takes a circuit C and converts it into a circuit C' consisting of m disjoint, ordered sub-computations $\text{Sub}_1, \dots, \text{Sub}_m$, where the input to sub-computation Sub_i depends only on the output of earlier sub-computations. Each of these sub-computations Sub_i is modeled as a non-uniform randomized poly-size circuit, with a “secret state.” It was proven that no information about the circuit C is leaked, even if each of these sub-computations is leaky. More specifically, the adversary can request to see a bounded-length function of each Sub_i (separately), and these leakage functions may be adaptively chosen.

These works also consider the continual leakage setting, where leakage occurs over and over again in time. In this setting, the secret state of each Sub_i must be continually updated or refreshed. To this end, after each computation, all the Sub_i ’s update their secret state by running a randomized protocol **Update**. We stress that leakage may occur during each of these update protocols, and that such leakage may be a function of both the current secret state and the randomness used by the **Update** procedure.

In this work, we use such a compiler which is secure in the LDS model [BCG⁺11]. The LDS model strengthens the OCL model in two ways. First, in the LDS model, the adversary is allowed to *view and control the entire communication between modules*; in contrast, the OCL model assumes the communication between modules is kept secret from the adversary, and that the messages are generated honestly. Second, in the LDS model, the adversary may leak adaptively on each module *in any order*. For instance, the adversary may leak a bit from Sub_i , then a bit from Sub_j , and based on the results, leak again on Sub_i . In contrast, the OCL model only allows the adversary to request leakage information from the module that is currently computing. In particular, this restricts the adversary to leak on modules in order (i.e., first leak from Sub_1 , then from Sub_2 , etc.).

REMARK 2.8. For the sake of simplicity of notation, we assume (without loss of generality) that the module Sub_i only sends messages to Sub_{i+1} (where we define $\text{Sub}_{m+1} \triangleq \text{Sub}_1$).

Moreover, we assume for simplicity that during each computation, where C is evaluated on some input v , each module Sub_i sends a single message to Sub_{i+1} , and that Sub_m does not send a message to any module, and simply outputs $C(v)$. This assumption indeed holds for the LDS compiler of [BCG⁺11] which is based on [GR12]. We note that this assumption is not needed for our result to be correct, but it simplifies the notation.

At the end of each time period, the modules “refresh” their inner state by applying a (possibly distributed) **Update** procedure, after which they erase their previous state. As with the rest of the computation, the **Update** procedure is also exposed to leakage, and the adversary controls the exchange of messages during the update.

DEFINITION 2.9 (LEAKY DISTRIBUTED SYSTEMS (LDS)). In a λ -bounded LDS attack, a PPT adversary \mathcal{A} interacts with modules $(\text{Sub}_1, \dots, \text{Sub}_m)$ by adaptively performing any sequence of the following actions:

- **Interact**(j, msg): For $j \in [m]$, send the message msg to the j 'th submodule, Sub_j , and receive the corresponding reply. Note that the modules are message-driven: they become activated when they receive a message from the attacker, at which point they compute and send the result, and then wait for additional messages.
- **Leak**(j, L): For $j \in [m]$ and a poly-size leakage function $L : \{0, 1\}^* \rightarrow \{0, 1\}$, if strictly fewer than λ queries of the form **Leak**(j, \cdot) have been made so far, \mathcal{A} receives the evaluation of L on the secret state of the j 'th submodule, Sub_j . Otherwise, \mathcal{A} receives \perp .

In a continual λ -LDS attack, the adversary \mathcal{A} repeats a λ -bounded LDS attack polynomially many times, where between every two consecutive attacks the secret states of the modules are updated. The update is done by running a distributed **Update** protocol among all the modules. We also allow \mathcal{A} to leak during the **Update** procedure, where the leakage function takes as input both the current secret state of Sub_j and the randomness it uses during the **Update** procedure.

We denote by time period t of submodule Sub_j the time period between the beginning of the $(t-1)$ 'st **Update** procedure and the end of the t 'th **Update** procedure in that submodule (note that these time periods are overlapping).⁵ We allow the adversary \mathcal{A} to leak at most λ bits from each Sub_j during each (local) time period.

We refer to such an adversary \mathcal{A} as an λ -LDS adversary, and denote the output of \mathcal{A} in such an attack by $\mathcal{A}[\lambda : \text{Sub}_1, \dots, \text{Sub}_m : \text{Update}]$.

We say that the collection of modules $(\text{Sub}_1, \dots, \text{Sub}_m)$ is λ -secure in the LDS model if for any λ -LDS adversary \mathcal{A} interacting with the modules as described above, there exists a PPT simulator who simulates the output of \mathcal{A} .

DEFINITION 2.10 (LDS-SECURE CIRCUIT COMPILER). We say that $(\mathcal{C}, \text{Update})$ is a λ -LDS secure circuit compiler if for any circuit C and $(\text{Sub}_1, \dots, \text{Sub}_m) \leftarrow \mathcal{C}(C)$, the following two properties hold:

⁵Intuitively, time period t is the entire time period where the t 'th updated secret states can be leaked. Note that during the t 'th **Update** procedure, both the $(t-1)$ 'st and the t 'th secret state may leak, which is why the time periods are overlapping.

1. **Correctness:** The collection of modules $(\text{Sub}_1, \dots, \text{Sub}_m)$ maintain the functionality of C when all the messages between them are delivered intact.
2. **Secrecy:** For every PPT λ -LDS adversary \mathcal{A} there exists a PPT simulator \mathcal{S} , such that for any ensemble of poly-size circuits $\{C_n\}$ and any auxiliary input $z \in \{0, 1\}^{\text{poly}(n)}$:

$$\left\{ \mathcal{A}(z)[\lambda : \text{Sub}_1, \dots, \text{Sub}_m : \text{Update}] \right\}_{n \in \mathbb{N}, C \in \mathcal{C}_n} \approx_c \left\{ \mathcal{S}^C(z, 1^{|C|}) \right\}_{n \in \mathbb{N}, C \in \mathcal{C}_n},$$

where \mathcal{S} only queries C on the inputs \mathcal{A} sends to the first module, Sub_1 .

THEOREM 2.11 ([BCG⁺11]). Assuming the existence of a non-committing encryption scheme and a λ -OCL circuit compiler which compiles a circuit C to $m(|C|)$ modules, there exists a λ -LDS secure circuit compiler $(\mathcal{C}, \text{Update})$ for which $\mathcal{C}(C)$ has the same number of modules, $m(|C|)$.

We note that there are known constructions of non-committing encryption schemes based on standard cryptographic assumptions, such as the DDH assumption and the RSA assumption. Moreover, a very recent work of Goldwasser and Rothblum [GR12] constructs a λ -OCL circuit compiler (unconditionally) with the following properties.

THEOREM 2.12 ([GR12]). For any security parameter k , there (unconditionally) exists a λ -OCL secure circuit compiler for $\lambda = \tilde{\Omega}(k)$, that takes any circuit C into a collection of $O(|C|)$ modules, each of size $O(k^3)$.

REMARK 2.13 (FOLKLORE). If one additionally assumes the existence of a fully homomorphic encryption (FHE) scheme, then there exists a λ -LDS secure circuit compiler $(\mathcal{C}, \text{Update})$ such that for every poly-size circuit C , the number of output sub-computations $\text{Sub}_1, \dots, \text{Sub}_m$ generated by \mathcal{C} is polynomial in the security parameter of the FHE scheme and independent of the size of C .

2.6 Weakly Leakage-Resilient MPC

Our construction of a leakage-resilient MPC protocol in the preprocessing model (defined in Section 3.2), uses as a building block an MPC protocol that is leakage-resilient with respect to a *weaker* notion of secrecy (where the ideal world is weakened), as was recently constructed in [BGG⁺11]. For lack of a better name, we call it *weakly* leakage-resilient MPC. Below, we recall the security model from [BGG⁺11].

Very briefly, the security definition in [BGG⁺11] follows the ideal/real world paradigm. They consider a real-world execution without a leak-free preprocessing stage, though they do assume the existence of an honestly generated CRS.⁶ The adversary, in addition to corrupting a number of parties, can obtain leakage information on the joint secret states of the honest parties at any point during the protocol execution. Leakage queries may be adaptively chosen based on all information received up to that point (including responses to previous leakage queries), and are computed on the joint secret states of all the honest parties.

⁶The CRS is simply a truly random string, and thus, could be generated in a leaky environment.

Note that one cannot hope to realize the standard ideal world security in the presence of such leakage attacks. To this end, [BGG⁺11] consider an ideal world experiment where in addition to learning the output of the function evaluation, the simulator is also allowed to request leakage on the inputs of all the honest parties jointly. Below, we describe the ideal and real world experiments and give the formal security definition from [BGG⁺11].

Ideal World.

We first describe the ideal world experiment, where n parties P_1, \dots, P_n interact with an ideal functionality for computing a function f . An adversary may corrupt any subset $M \subset \mathcal{P}$ of the parties. As in the standard MPC ideal world experiment, the parties send their inputs to the ideal functionality and receive the output of f evaluated on all inputs. The main difference from the standard ideal world experiment is that the adversary is also allowed to make *leakage queries* on the inputs of the honest parties. Such queries are evaluated on the *joint* collection of all parties' inputs. The ideal world execution proceeds as follows.

Inputs: Each party P_i obtains an input x_i . The adversary is given auxiliary input z and selects a subset of parties $M \subset \mathcal{P}$ to corrupt.

Sending inputs to trusted party: Each honest party P_i sends its input x_i to the ideal functionality. For each corrupted party $P_i \in M$, the adversary may select any value x'_i and send it to the ideal functionality.

Trusted party computes output: Let x'_1, \dots, x'_n be the inputs that were sent to the ideal functionality. The ideal functionality computes $f(x'_1, \dots, x'_n)$.

Adversary learns output: The ideal functionality first sends the evaluation $f(x'_1, \dots, x'_n)$ to the adversary. The adversary replies with either *continue* or *abort*.

Honest parties learn output: If the message is *abort*, the ideal functionality sends \perp to all honest parties. If the adversary's message was *continue*, then the ideal functionality sends the function evaluation $f(x'_1, \dots, x'_n)$ to all honest parties.

Leakage queries on inputs: The adversary may send (adaptively chosen) leakage queries in the form of efficiently computable functions L_j (described as a circuit). On receiving such a query, the ideal functionality computes $L_j(x'_1, \dots, x'_n)$ and returns the output to the adversary.

Outputs: Honest parties output their inputs and the messages they obtained from the ideal functionality. Malicious parties may output an arbitrary PPT function of their initial input (auxiliary input and random-tape) and the message it has obtained from the ideal functionality.

An ideal world adversary \mathcal{S} who obtains a total of λ bits of leakage is referred to as a λ -leakage ideal adversary. The overall output of the ideal-world experiment consists of all the inputs and values received by honest parties from the ideal functionality, together with the output of the adversary, and is denoted by $\text{W-IDEAL}_{\mathcal{S}, M}^f(1^k, \vec{x}, z)$.

Real World.

The real-world experiment begins by first choosing a common random string crs . Then, each party P_i receives an input x_i and the adversary \mathcal{A} receives auxiliary input z . These values can depend arbitrarily on the crs , but need to be efficiently computable given the crs . However, for the sake of simplicity of notation, throughout this section we assume that these values are independent of the crs .

The adversary \mathcal{A} selects any arbitrary subset $M \subset \mathcal{P}$ of the parties to corrupt. Each corrupted party $P_i \in M$ hands over its input to \mathcal{A} . The parties P_1, \dots, P_n now engage in an execution of a real n -party protocol Π . The adversary \mathcal{A} sends all messages on behalf of the corrupted parties, and may follow an arbitrary polynomial-time strategy. In contrast, the honest parties follow the instructions of Π . Furthermore, at any point during the protocol execution, the adversary may make leakage queries of the form L and learn $L(\text{state}_{\mathcal{P} \setminus M})$, where $\text{state}_{\mathcal{P} \setminus M}$ denotes the concatenation of the protocol states state_i of each honest party P_i . We allow the adversary to choose the leakage queries adaptively.

Honest parties have the ability to toss fresh coins at any point in the protocol, and at that point these coins are added to the state of that party. At the conclusion of the protocol execution, each honest party P_i generates an output according to Π . Malicious parties may output an arbitrary PPT function of the view of \mathcal{A} .

An adversary \mathcal{A} who obtains at most λ bits of leakage is referred to as a λ -leakage real adversary. Let Gen_w denote the CRS generation algorithm. Further, let $\text{W-REAL}_{\mathcal{A}}^\Pi(1^k, \text{crs}, \vec{x}, z)$ be the random variable that denotes the values output by the parties at the end of the protocol Π (using $\text{crs} \leftarrow \text{Gen}_w(1^k)$ as the CRS). Then, the overall output of the real-world experiment is defined as the tuple $(\text{crs}, \text{W-REAL}_{\mathcal{A}, M}^\Pi(1^k, \text{crs}, \vec{x}, z))$.

We now state the formal security definition.

DEFINITION 2.14 (λ -WEAKLY LEAKAGE-RESILIENT MPC).

A protocol Π evaluating a functionality f is a λ -weakly leakage-resilient MPC protocol if for every PPT λ -leakage real adversary \mathcal{A} , there exists a λ -leakage ideal adversary $\mathcal{S} = (\mathcal{S}^{\text{crs}}, \mathcal{S}^{\text{exec}})$, corrupting the same parties as \mathcal{A} , such that for every input vector \vec{x} , every auxiliary input $z \in \{0, 1\}^*$, and every subset $M \subset \mathcal{P}$, it holds that the distribution

$$\left\{ \text{crs}, \text{W-IDEAL}_{\mathcal{S}^{\text{exec}}(\text{crs}, \text{trap}), M}^f(1^k, \vec{x}, z) \right\}_{k \in \mathbb{N}}$$

is computationally indistinguishable from the distribution

$$\left\{ \text{crs}', \text{W-REAL}_{\mathcal{A}, M}^\Pi(1^k, \text{crs}', \vec{x}, z) \right\}_{k \in \mathbb{N}},$$

where $(\text{crs}, \text{trap}) \leftarrow \mathcal{S}^{\text{crs}}(1^k)$, and $\text{crs}' \leftarrow \text{Gen}_w(1^k)$.

THEOREM 2.15 ([BGG⁺11]). *Based on the DDH assumption, for every poly-size function f , for every leakage bound $\lambda \in \mathbb{N}$, and any number of parties and corrupted parties, there exists a protocol Π in the common random string model for computing f that is λ -weakly leakage resilient as per Definition 2.14.*

REMARK 2.16. *We note that Theorem 2.15 holds even if we allow the input vector \vec{x} and the auxiliary input z to be arbitrary poly-time computable functions of the crs . We eliminated this dependency from Definition 2.14 only for the sake of simplicity of notation.*

REMARK 2.17 (STANDALONE VS. UC SECURITY). *The main result in [BGG⁺11] actually achieves a stronger notion of universally composable (UC) security, at the cost of additionally relying on the decisional linear assumption over bilinear groups. Indeed, their UC-secure WLR-MPC construction relies on a leakage-resilient UC-NIZK system, whose only known construction [GJS11, GOS06] is based on the decisional linear assumption in the bilinear groups setting.*

However, for the present paper, it suffices to obtain a “standalone” secure construction of WLR-MPC. Thus, it is possible to replace the UC-NIZK system with a standalone secure interactive weakly leakage-resilient ZKPoK system. This, in turn, can be based on the DDH assumption. The resulting WLR-MPC achieves standalone security based on only the DDH assumption in the CRS model.

Security against disjoint leakage.

In Definition 2.14, the real-world adversary \mathcal{A} is allowed to obtain *joint* leakage on the secret states of the honest parties. In the present work, we consider a weaker adversarial model, in which the leakage on each honest party in the real world is *disjoint* (i.e., \mathcal{A} is not allowed to leak on the joint secret states of the honest parties). Theorem 2.15 clearly still applies to this setting. However, we note that the *ideal world* guarantee does not become stronger when we consider this set of restricted adversaries: that is, even to simulate such adversaries, the simulator \mathcal{S} needs *joint* leakage on the inputs of all the honest parties.⁷

2.6.1 Security for randomized functions

We note that Theorem 2.15 holds for *deterministic* functions. In this work, we need to use a weak leakage resilient protocol for *randomized* functions (since the modules in the OCL leakage resilient circuit compute randomized functions). We show that in our setting, where leakage in the real world is disjoint, the number of parties is polynomially related to the security parameter, and a constant fraction of the parties are honest, then we can construct weak leakage resilient protocols for randomized functions.

THEOREM 2.18 (INFORMAL). *Theorem 2.15 holds also for randomized functions if we restrict the adversaries to leak on the honest parties disjointly, when the number of parties is polynomially related to the security parameter, and ϵ -fraction of them are honest for some constant $\epsilon > 0$.*

Due to lack of space we defer the proof of this theorem to the final version.

3. OUR MODEL

In this section, we present the MPC model and the security definition considered in this paper. We start by giving a brief overview of our model and then proceed with a formal description.

Overview. We consider the setting of n parties $\mathcal{P} = \{P_1, \dots, P_n\}$ within a synchronous point-to-point network with authenticated broadcast channel [DS83] who wish to jointly compute

⁷As mentioned in Section 1.4, if we could simulate real-world adversaries that obtain only *disjoint* leakage queries, with a simulator that obtains only *disjoint* leakage queries, then this would almost immediately give us a result similar to ours: An MPC protocol with preprocessing that is secure against continual leakage.

any ppt function over their private inputs. Specifically, we consider the case where the parties wish to perform *arbitrarily many* evaluations of functions of their choice. We refer to a protocol that allows computation of multiple functions (over a given set of inputs) as a *multi-function MPC protocol*. Unlike the standard MPC setting, we consider security of a multi-function MPC protocol against “leaky” adversaries that may (continuously) leak on the secret state of each honest party during the protocol execution.

To formally define security, we turn to the real/ideal paradigm. Very briefly, we consider a real-world execution where an adversary, who corrupts any arbitrary number of parties in the system, may additionally obtain arbitrary bounded, independent leakage on the secret state of each honest party. However, unlike the recent works on leakage-resilient interactive protocols [GJS11, BCH11, BGK11, DHP11, BGG⁺11], we consider the *standard* ideal world model, where the adversary does not learn *any* information on the honest party inputs.

Note that if we do not put any restriction on the real-world adversary, and in particular, if he is allowed to obtain leakage *throughout* the protocol execution, then it is impossible to realize the standard ideal world model, since the adversary may simply leak on the inputs of the honest parties, while this information cannot be simulated in the ideal world. With this in mind, we (necessarily) allow for a “leak-free” one-time *preprocessing stage* that happens at the beginning of the real-world execution. Furthermore, to withstand *continual* leakage attacks, we (necessarily) allow for periodic *updates* of the secret values of the parties. We allow leakage to occur during this update procedure as usual.

We now proceed to give a formal description of our model in the remainder of this section. In Section 3.1, we describe the ideal world experiment. In Section 3.2, we describe the real world experiment. Finally, in Section 3.3, we present our security definition.

Throughout this work, we assume that the functions to be evaluated give the same output to all parties. This is for simplicity of exposition, since otherwise, if the output itself is a secret value (given to an honest party) then this value can be leaked. This can be handled by complicating our security guarantees, and, indeed, one can tweak our construction to ensure that the adversary learns *only* leakage information on such outputs. However, for the sake of simplicity, we choose to avoid this issue in this manuscript.

3.1 Ideal World

In the ideal world, each party P_i sends her input x_i to a trusted third party. Whenever the adversary \mathcal{A} sends a poly-size circuit f to the trusted party, it sends back $f(x_1, \dots, x_n)$. Since we consider the case of dishonest majority, we can only obtain security with abort: i.e., the adversary first receives the function output $f(x_1, \dots, x_n)$, and then chooses whether the honest parties also learn the output, or to prematurely abort. The adversary can query the trusted party many times with various functions f_j . Moreover, these functions can be adaptively chosen, based on the outputs of previous functions. The ideal world model is formally described below.

Inputs: Each party P_i obtains an input x_i . The adversary is given auxiliary input z . He selects a subset of the

parties $M \subset \mathcal{P}$ to corrupt, and is given the inputs x_ℓ of each party $P_\ell \in M$.

Sending inputs to trusted party: Each honest party P_i sends its input x_i to the ideal functionality. For each corrupted party $P_i \in M$, the adversary may select any value x'_i and send it to the ideal functionality.

Trusted party computes output: Let x'_1, \dots, x'_n be the inputs that were sent to the trusted party. Then, the following is repeated for any (unbounded) polynomial number of times:

- **Function selection:** The adversary chooses a poly-size circuit f_j , and sends it to the ideal functionality.
- **Adversary learns output:** The ideal functionality sends the evaluation $f_j(x'_1, \dots, x'_n)$ to the adversary. The adversary replies with either **continue** or **abort**.
- **Honest parties learn output:** If the adversary's message was **abort**, then the trusted party sends \perp to all honest parties and the experiment concludes. Otherwise, if the adversary's message was **continue**, then it sends the function output $f_j(x'_1, \dots, x'_n)$ to all honest parties.

Outputs: Honest parties output all the messages they obtained from the ideal functionality. Malicious parties may output an arbitrary PPT function of the adversary's view.

The overall output of the ideal-world experiment consists of the outputs of all parties. For any ideal-world adversary \mathcal{S} with auxiliary input $z \in \{0, 1\}^*$, any input vector \vec{x} , any set of functions $\{f_j\}_{j=1}^p$ chosen by the adversary, and security parameter k , we denote the output of the corresponding ideal-world experiment by

$$\text{IDEAL}_{\mathcal{S}, M}(1^k, \vec{x}, z, \{f_j\}_{j=1}^p).$$

Note that this is a slight abuse of notation since the functions $\{f_j\}_{j=1}^p$ may be chosen adaptively.

3.2 Real World

The real world execution begins by an adversary \mathcal{A} selecting any arbitrary subset of parties $M \subset \mathcal{P}$ to corrupt. The parties then engage in an execution of a real n -party multi-function MPC protocol $\Pi = (\Pi_{\text{Pre}}, \Pi_{\text{Input}}, \Pi_{\text{Online}})$ that consists of three stages, namely, (a) a *preprocessing phase*, (b) an *input phase*, and (c) an *online phase*, as described below. We assume that honest parties have the ability to toss fresh coins at any point. Throughout the execution of Π , the adversary \mathcal{A} sends all messages on behalf of the corrupted parties, and may follow an arbitrary polynomial-time strategy. In contrast, the honest parties follow the instructions of Π . Furthermore, at any point during the protocol execution (except during the preprocessing and the input phases), the adversary may leak on the *entire* secret state of each honest parties, via an *MPC leakage query*, defined as follows.

DEFINITION 3.1. An MPC leakage query is defined by $\text{Leak}(i, L)$, where $i \in [n]$ and $L : \{0, 1\}^* \rightarrow \{0, 1\}$ is a poly-size circuit. When an adversary sends a leakage query $\text{Leak}(i, L)$, he receives the evaluation of L on the entire secret state of party P_i .

We now formally describe the different phases in the protocol.

Preprocessing phase: This phase is *interactive* and *leak-free*, and is run only once. It is independent of the inputs of the parties, and is independent of the functions that will later be evaluated. Thus, this phase can be run in the beginning of time, before the parties even know what their inputs are, or what functions they would like to evaluate.

We assume that no leakage occurs during the run of this preprocessing phase, but we do allow leakage to occur as soon as the preprocessing phase ends. At the end of this phase each party P_i has an (initial) secret state $\text{state}_1^{P_i}$.

Input phase: This phase is *non-interactive* and *leak-free*, and depends only on the inputs x_1, \dots, x_n (independent of the functions to be computed). Whenever a party P_i gets (or chooses) a secret input x_i , she does some local computation which may depend on her secret input x_i and on her secret state $\text{state}_1^{P_i}$. She then sends a message to all parties, and erases her secret input x_i . One may think of this as a “hot potato” model, where the parties never store their inputs for very long (since they are concerned with leakage), but rather immediately share their input as if it were a “hot potato”.

We assume that the party P_i is not leaked upon during the execution of this phase. However, leakage may occur between the preprocessing phase and the input phase, and leakage may occur immediately after the input phase.

We emphasize that each party can change her input as often as she wants by simply re-running the input phase with the new input.⁸

Online phase: This phase takes place in a *leaky* environment. During this phase, the parties carry out an unbounded number of function evaluations on their inputs, and update their respective secret states. At any point during this phase, \mathcal{A} may make adaptively-chosen leakage queries, as per Definition 3.1, in the manner as described below.

Whenever \mathcal{A} wishes to compute a function f_j (represented as a poly-size circuit), all parties execute the function evaluation protocol Π_{Comp} , described below. Whenever \mathcal{A} wants the honest parties to update their secret states, all parties execute the update protocol Π_{Update} , described below. We let $\Pi_{\text{Online}} = (\Pi_{\text{Comp}}, \Pi_{\text{Update}})$. We begin at leakage time period $\ell = 1$; after each update procedure, ℓ is incremented.

• Computation procedure:

1. All parties execute protocol $\Pi_{\text{Comp}}(f_j)$, where honest parties P_i act in accordance with input $\text{state}_\ell^{P_i}$. Note that the secret state of parties may change during the execution of this protocol, as dictated by Π_{Comp} .

⁸For simplicity, in the security proof in Section 5, we assume that the parties run the input phase only once, however the proof extends readily to the case that the parties rerun the input phase many times with different inputs.

2. At the conclusion of the computation phase, each honest party P_i outputs his final message of the protocol (which should correspond to the evaluation of f_j). Malicious parties may output an arbitrary PPT function of the view of \mathcal{A} .

• ℓ^{th} **Update procedure:**

1. All parties execute protocol Π_{Update} , where honest parties P_i act in accordance with input $\text{state}_{\ell}^{P_i}$.
2. At the conclusion of the update phase, each honest party P_i sets $\text{state}_{\ell+1}^{P_i}$ to be P_i 's output from Π_{Update} . Each honest P_i erases $\text{state}_{\ell}^{P_i}$.
3. Increment $\ell \leftarrow \ell + 1$.

Leakage: Initialize each leaked_{ℓ} to 0. Each leakage query (i, L) made by \mathcal{A} during the ℓ^{th} time period is answered as follows.

- During the **computation** phase: if $\text{leaked}_{\ell} \geq \lambda$, then \mathcal{A} receives \emptyset . Otherwise, \mathcal{A} receives the evaluation of L on the current secret state of party P_i , and $\text{leaked}_{\ell} \leftarrow \text{leaked}_{\ell} + 1$.
- In ℓ^{th} **update** phase: if *either* $\text{leaked}_{\ell} \geq \lambda$ or $\text{leaked}_{\ell+1} \geq \lambda$, then \mathcal{A} receives \emptyset . Otherwise, \mathcal{A} receives the evaluation of L on the current secret state of party P_i , and both $\text{leaked}_{\ell} \leftarrow \text{leaked}_{\ell} + 1$ and $\text{leaked}_{\ell+1} \leftarrow \text{leaked}_{\ell+1} + 1$.

We emphasize that the \mathcal{A} 's leakage queries may be made on any party, adaptively chosen based on all information received up to that point (including responses to previous leakage queries). The only restriction is that the number of bits leaked between the execution of any two consecutive update protocols is bounded. Note that the leakage queries made during the ℓ^{th} update phase (where parties transition between their ℓ^{th} and $(\ell+1)^{\text{st}}$ secret states) are counted against *both* the ℓ^{th} and $(\ell+1)^{\text{st}}$ time period, where the ℓ^{th} time period is the time period where the party stores her ℓ^{th} secret state. The reason for this “double counting” is that during the ℓ^{th} update phase, the adversary can leak both on the ℓ^{th} secret state and on the $\ell+1^{\text{st}}$ secret state of the party.

We refer to an adversary who corrupts t parties $M \subset \mathcal{P}$ and makes up to λ leakage queries in each time period as a (t, λ) -*continual leakage adversary*.

For any adversary \mathcal{A} with auxiliary input $z \in \{0, 1\}^*$, any inputs $\{x_i\}_{i=1}^n$, any set of functions $\{f_j\}_{j=1}^p$ chosen (adaptively) by the adversary, and any security parameter k , we denote the output of the multi-function MPC protocol $\Pi = (\Pi_{\text{Pre}}, \Pi_{\text{input}}, \Pi_{\text{Online}})$ by

$$\text{REAL}_{\mathcal{A}, M}^{\Pi} \left(1^k, \vec{x}, z, \{f_j\}_{j=1}^p \right).$$

Loosely speaking, we say that a protocol Π is a *leakage-resilient* multi-function MPC protocol if any adversary, who corrupts a subset of parties, receives leakage information as described above, and runs the protocol with honest parties on any (unbounded) sequence of functions f_1, \dots, f_p , gains *no information* about the inputs of the honest parties beyond the output of the functions $f_j(x_1, \dots, x_n)$ for $j = 1, \dots, p$. We formalize this in the next subsection.

3.3 Security Definition

In what follows, we formally define our model of security; i.e., what it means for a real-world protocol to emulate the desired ideal world.

DEFINITION 3.2 (LEAKAGE-RESILIENT MPC). *A multi-function evaluation protocol $\Pi = (\Pi_{\text{Pre}}, \Pi_{\text{input}}, \Pi_{\text{Online}})$ is said to be λ -leakage-resilient against t malicious parties if for every PPT (t, λ) -continual leakage MPC adversary \mathcal{A} in the real world, there exists a PPT adversary \mathcal{S} corrupting the same parties in the ideal world such that for every input vector \vec{x} , every auxiliary input z , and any (adaptively chosen) set of functions $\{f_j\}_{j=1}^p$ where $p = \text{poly}(k)$, it holds that*

$$\text{IDEAL}_{\mathcal{S}, M} \left(1^k, \vec{x}, z, \{f_j\}_{j=1}^p \right) \approx_c \text{REAL}_{\mathcal{A}, M}^{\Pi} \left(1^k, \vec{x}, z, \{f_j\}_{j=1}^p \right).$$

Note that we do *not* allow the simulator to request leakage on honest parties' inputs in the ideal world, as was done in [BCH11, DHP11, BGG⁺11], and thus model a stronger notion of secrecy than what was achieved in prior works.⁹

4. OUR CONSTRUCTION

In this section, we construct a leakage-resilient multi-function MPC protocol, as defined in Section 3. Our construction uses the following ingredients:

1. (\mathcal{C} , Update): a λ -LDS secure circuit compiler, as in Theorem 2.11. Recall for a circuit C , the compiler $\mathcal{C} : C \mapsto (\text{Sub}_1, \dots, \text{Sub}_m)$ yields a collection of modules whose sequential execution evaluates C , and which are secure in the LDS model (see Section 2.5 for details).
2. Elect: a public-coin protocol for electing m disjoint committees (where m is the number of modules from above), each of size approximately k , as in Lemma 2.6.
3. (Gen_{eq} , Com, Rec, $\mathcal{S}_{\text{eq}} = (\mathcal{S}_{\text{eq}}^{\text{crs}}, \mathcal{S}_{\text{eq}}^{\text{com}})$): a crs-based equivocal commitment scheme, as in Lemma 2.5.
4. (Gen , Enc, Dec, Eval): a fully homomorphic public-key encryption (FHE) scheme that is certifiable with respect to an efficiently testable set $R \subseteq \{0, 1\}^{\text{poly}(k)}$, as described in Section 2.4.
5. (Gen_{nizk} , P, V, $\mathcal{S}_{\text{nizk}} = (\mathcal{S}_{\text{nizk}}^{\text{crs}}, \mathcal{S}_{\text{nizk}}^{\text{proof}})$): a non-interactive zero-knowledge (NIZK) proof of knowledge (as in Lemma 2.3) for the NP language

$$L = \{(\text{pk}, \hat{x}) : \exists (x, r) \text{ s.t. } r \in R, \hat{x} = \text{Enc}_{\text{pk}}(x; r)\}, \quad (1)$$

where $R \subseteq \{0, 1\}^{\text{poly}(k)}$ is the set for which the FHE scheme is certifiable.

6. MPC(F): a standard multiparty computation protocol for evaluating a function F , with no leakage resilience guarantees, such as [GMW87].
7. (Gen_w , MPC $_w(F)$): a λ -*weakly* leakage-resilient multiparty computation (WLR-MPC) protocol for evaluating a function F in the common random string model, as given by Theorem 2.18.

⁹With the (necessary) addition of a one-time leak-free preprocessing phase.

THEOREM 4.1. *Fix any constants $\epsilon, \delta > 0$. Then, assuming the existence of the ingredients 1 - 7 listed above (where the LDS circuit compiler and WLR-MPC protocol are secure with leakage parameter λ), there exists a λ -leakage-resilient multi-function evaluation MPC protocol $\Pi = (\Pi_{\text{Pre}}, \Pi_{\text{Input}}, \Pi_{\text{Update}})$ for $n \geq k^\delta$ parties, tolerating $t = (1 - \epsilon)n$ corrupted parties.*

Remark.

The reason we need the number of parties to be polynomially related to the security parameter is two-fold. First, in the preprocessing phase, the protocol Π_{Pre} elects committees $\mathcal{E}_1, \dots, \mathcal{E}_m$, and security of the protocol relies on the fact that these committees are disjoint and each committee contains a constant fraction of honest parties. Thus, if n is a constant, then the resulting security guarantee is that the advantage of any PPT distinguisher in the security game is bounded (from below) by a constant. More generally, the advantage is $\geq 2^{-\epsilon n}$ (see Lemma 2.6).

The second reason we the number of parties must be large is that the number m of disjoint committees $\mathcal{E}_1, \dots, \mathcal{E}_m$ we need to elect is large. This is because the number of committees is exactly the number of modules generated by the LDS compiler, when applied to the decryption circuit Dec_{sk} of the underlying FHE scheme. Since the only LDS compiler we know (that does not use secure hardware) requires $m = O(|\text{Dec}_{\text{sk}}|)$, the number of modules must be at least the security parameter of the underlying FHE scheme (which we can set to be k^δ).

We now present the protocol $\Pi = (\Pi_{\text{Pre}}, \Pi_{\text{Input}}, \Pi_{\text{Online}})$, where $\Pi_{\text{Online}} = (\Pi_{\text{Comp}}, \Pi_{\text{Update}})$. At a high level, Π is defined as follows:

Preprocessing phase Π_{Pre} : In the preprocessing phase, the parties run a (standard) MPC to collectively generate a key pair (pk, sk) for the FHE scheme, and to secret share sk in such a way that (a) learning the shares of corrupted parties, and leakage on each remaining share, does not damage the security of the FHE, but (b) collectively, the shares can be used to evaluate the decryption circuit in a leaky environment. More specifically, shares are generated by running the LDS compiler on the decryption circuit $\text{Dec}_{\text{sk}}(\cdot)$ (with sk hardwired) to obtain a sequence of modules $\text{Sub}_1, \dots, \text{Sub}_m$; the parties elect corresponding (disjoint) committees $\mathcal{E}_1, \dots, \mathcal{E}_m$, and secret share each Sub_j among parties in \mathcal{E}_j , using a standard secret sharing scheme (e.g., the simple xor scheme). To ensure that parties provide the correct secret shares of the Sub_j 's in future computations, within the MPC the parties collectively generate and publish commitments to each correct share.

In addition, the preprocessing phase is used to generate crs setup information for subsidiary tools used throughout the protocol. This is also done via a (standard) MPC.

(Note that the preprocessing procedure is independent of parties' secret inputs and functions to be evaluated.)

Input phase Π_{Input} : Each time a party P_i wishes to submit a new secret input x_i , she computes and publishes an encryption \hat{x}_i of x_i under the FHE scheme (specifically, under the public key pk for the FHE that was

generated during the preprocessing phase). To ensure that malicious parties do not send malformed ciphertexts, which could ruin the correctness of homomorphic evaluation later down the line (and potentially damage security), each party accompanies her published ciphertext \hat{x}_i with a NIZK proof of knowledge that the ciphertext is properly formed.

Online phase Π_{Online} : The online phase consists of two parts: the *computation phase*, in which parties collectively evaluate a queried function f on all inputs, and the *update phase*, in which parties collectively refresh their secret states.

Computation phase Π_{Comp} : Each time the adversary requests the evaluation of a function f on all parties' inputs, two steps take place. First, each party (individually) homomorphically evaluates the function f on the *encrypted* vector of inputs $\hat{x} = (\hat{x}_1, \dots, \hat{x}_n)$. Note that the result, \hat{y}_f , is an encryption of the desired value $f(\vec{x})$. Next, the parties jointly decrypt, using their shares of sk from the preprocessing phase. Namely, the parties execute the sequence of modules $\text{Sub}_1, \dots, \text{Sub}_m$ obtained by the LDS compiler applied to $\text{Dec}_{\text{sk}}(\cdot)$, where the input to the first module Sub_1 is \hat{y}_f . To emulate the execution of each module Sub_j , the parties of committee \mathcal{E}_j run a WLR-MPC protocol among themselves. Within the WLR-MPC, the parties of \mathcal{E}_j combine their secret shares $\text{Sub}_{j,i}$ (checking first to make sure each party's share agrees with the corresponding published commitment) and execute the computation dictated by Sub_j . Communication between modules is performed by having *all* parties of committee \mathcal{E}_j send the appropriate message to *all* parties of the next committee, \mathcal{E}_{j+1} . The output of the final module, Sub_m , is the evaluation $f(\vec{x})$.

Update phase Π_{Update} : Each time the adversary requests that parties update their secret states, the parties execute the update procedure of the LDS compiler, where each module computation is performed via a WLR-MPC among the parties of the corresponding committee, as above. The only difference here is that the secret state Sub_j of each module is also changing. Thus, during each execution of a module Sub_j , the corresponding committee must also generate fresh secret shares for its parties, and new commitment and decommitment information for each share. To provide the required correctness and secrecy guarantees, this process takes place as part of the committee's WLR-MPC execution.

The formal descriptions of Π_{Pre} , Π_{Input} , Π_{Comp} , and Π_{Update} appear in Figures 1, 2, 3, and 4, respectively.

REMARK 4.2. *Throughout the protocol description (as well as throughout the proof), we define **abort** to be the action of broadcasting the message "abort" to all parties. At any point in which a party receives an "abort" message, he runs **abort** and exits the protocol.*

5. PROOF OF SECURITY

PROOF OF THEOREM 4.1. Let \mathcal{A} be any real-world PPT adversary for Π . Denote by $M \subset \mathcal{P}$ the set of parties corrupted by \mathcal{A} .

Preprocessing Phase:

Input: 1^k . No leakage allowed.

1. The parties elect m disjoint committees \mathcal{E}_j of size approximately k' by running **Elect**. Here, k' is the security parameter for the FHE scheme and $m = \text{poly}(k')$ is the number of modules produced by the LDS compiler when run on the decryption circuit Dec_{sk} for this security parameter. We take $k' = k^{\Theta(1)}$ as large as possible while maintaining $m \cdot k'^2 \leq n$.
2. All parties engage in an execution of the (standard) MPC protocol $\text{MPC}(F_{\text{crs}})$ to compute the (randomized) functionality F_{crs} described as follows. Functionality F_{crs} does not take any inputs and computes the following: (a) a CRS $\text{crs}_w \leftarrow \text{Gen}_w(1^k)$ for the weakly leakage-resilient MPC protocol $(\text{Gen}_w, \text{MPC}_w(F))$, (b) a CRS $\text{crs}_{\text{eq}}^i \leftarrow \text{Gen}_{\text{eq}}(1^k)$ for each party P_i for the equivocal commitment scheme $(\text{Gen}_{\text{eq}}, \text{Com}, \text{Rec}, \mathcal{S}_{\text{eq}})$, and (c) a CRS $\text{crs}_{\text{nizk}}^i \leftarrow \text{Gen}_{\text{nizk}}(1^k)$ for each party P_i for the NIZK proof of knowledge system $(\text{Gen}_{\text{nizk}}, \text{P}, \text{V}, \mathcal{S}_{\text{nizk}})$. Denote by crs the tuple $(\{\text{crs}_{\text{eq}}^i, \text{crs}_{\text{nizk}}^i\}_{i=1}^n, \text{crs}_w)$.
3. All parties engage in an execution of the (standard) MPC protocol $\text{MPC}(F_{\mathcal{E}_1, \dots, \mathcal{E}_m, \text{crs}})$ to collectively compute the randomized functionality $F_{\mathcal{E}_1, \dots, \mathcal{E}_m, \text{crs}}$ (that does not take any inputs) defined as follows:

The (randomized) function:

Generate a key pair $(\text{sk}, \text{pk}) \leftarrow \text{Gen}(1^{k'})$ for the FHE scheme.

Evaluate the LDS circuit transformation on the decryption circuit for sk :

$$(\text{Sub}_1, \dots, \text{Sub}_m) \leftarrow \mathcal{C}(\text{Dec}_{\text{sk}}).$$

(We abuse notation and denote by Sub_j both the computation of the submodule and the secret state corresponding to the submodule.)

For each $j \in [m]$, secret share $\text{Sub}_j = \text{Sub}_{j,1} \oplus \dots \oplus \text{Sub}_{j,|\mathcal{E}_j|}$ among the parties in the j 'th committee, \mathcal{E}_j .

For each share $\text{Sub}_{j,i}$ generated in the previous step, compute a commitment $(c_{j,i}, d_{j,i}) \leftarrow \text{Com}(\text{crs}_{\text{eq}}^\alpha, \text{Sub}_{j,i})$, where P_α is the i 'th party in \mathcal{E}_j (i.e., the party that receives the share $\text{Sub}_{j,i}$).

Output: The outputs are as follows.

All parties: $\text{pk}, \{c_{j,i}\}_{j \in [m], i \in [|\mathcal{E}_j|]}$
 Party i of \mathcal{E}_j : $\text{Sub}_{j,i}, d_{j,i}$

4. Each party erases all intermediate values of the MPC executions.

(Note that Steps 2 and 3 can be combined into a single multi-party computation execution, but have been split into two separate executions for ease of explanation and proof).

Figure 1: Protocol Π_{pre} : Preprocessing phase.

Input Phase: Party P_i wishes to submit a new private input, x_i . No leakage allowed.

Public inputs: $\text{pk}, \{\text{crs}_{\text{nizk}}^i\}_{i=1}^n$.

Private input: x_i , held by party P_i .

Party P_i performs the following steps:

1. Sample a value $r_i \leftarrow R \subseteq \{0,1\}^{\text{poly}(k)}$ via rejection sampling. Recall the FHE scheme is certifiable with respect to the set $R \subseteq \{0,1\}^{\text{poly}(k)}$ (see Definition 2.7).
2. Encrypt $\hat{x}_i = \text{Enc}_{\text{pk}}(x_i; r_i)$.
3. Compute a NIZK proof of knowledge that $(\text{pk}, \hat{x}_i) \in L$ using witness (x_i, r_i) and CRS $\text{crs}_{\text{nizk}}^i$. (See Equation (1) above for the definition of L). That is, $\pi_i \leftarrow \text{P}(\text{crs}_{\text{nizk}}^i, (\text{pk}, \hat{x}_i), (x_i, r_i))$.
4. Send the pair (\hat{x}_i, π_i) to all parties.
(It suffices to send it to parties in \mathcal{E}_1 .)
5. Erase initial input x_i , together with all intermediate values of the input phase.

Figure 2: Protocol Π_{input} : Input phase.

We construct an adversary \mathcal{S} in the ideal world who simulates the real-world view of \mathcal{A} by simulating the honest parties in the real world experiment. We do so by a sequence of intermediate steps, where we show how to simulate these values given less and less information, eventually given only the function evaluations $f(x_1, \dots, x_n)$, as in the ideal-world experiment. More explicitly, we consider the following sequence of hybrid experiments. We note that all ideal functionalities in the hybrid experiments are implicitly *with abort*: i.e., the ideal functionality first outputs to only the adversary, who decides whether outputs are also delivered to honest parties, or whether the protocol ends in *abort*.

In what follows we describe all of our hybrid experiments. We defer the construction of the corresponding simulator and the proof of indistinguishability to the full version. For each hybrid, we include (in the parentheses) the primary reason why Hybrid i can be simulated from Hybrid $i - 1$.

Hybrid 0. The real world: i.e., the adversary interacts with honest parties in the real-world experiment running Π .

Hybrid 1. (Elect protocol) The same as the real-world experiment, except that if any of the committees $\mathcal{E}_1, \dots, \mathcal{E}_m$ elected during the preprocessing phase has *fewer* than $\frac{\epsilon}{2}k$ parties, or if the fraction of honest parties in any committee is less than $\frac{\epsilon}{3}$, the experiment immediately concludes with output fail. We assume for simplicity of notation (later on) that, if the experiment does not fail, the first party of each committee \mathcal{E}_j is honest.

Hybrid 2. (MPC security) The same as Hybrid 1, except instead of collectively generating the CRS values (for the equivocal commitment scheme, the WLR-MPC, and the NIZK proof system) via an MPC protocol during the preprocessing phase, we assume a setup model where these values are (honestly) generated beforehand, and all parties run with these CRS values

Computation Phase:

Public inputs: $f, \text{pk}, \hat{x} = (\text{Enc}_{\text{pk}}(x_1), \dots, \text{Enc}_{\text{pk}}(x_n))$, $\text{crs} = (\{\text{crs}_{\text{eq}}^i, \text{crs}_{\text{nizk}}^i\}_{i=1}^n, \text{crs}_w)$, $\mathcal{E}_1, \dots, \mathcal{E}_m$, $\{c_{j,i}\}_{j \in [m], i \in [|\mathcal{E}_j|]}$.
 Private inputs: $(\text{Sub}_{j,i}, d_{j,i})$, held by party i of \mathcal{E}_j .

1. All parties homomorphically evaluate f on the encrypted input vector: $\hat{y} = \text{Eval}_{\text{pk}}(\hat{x}, f)$.
 (It suffices that only parties in \mathcal{E}_1 compute \hat{y} .)
2. The parties execute the Decryption Cascade with $\text{input}_1 = \hat{y}$.

Decryption Cascade:

1. For $j = 1, \dots, m$:

The parties in \mathcal{E}_j engage in an execution of the λ -weakly leakage-resilient MPC protocol $\text{MPC}_w(F_j)$ using CRS crs_w to compute the (randomized) functionality F_j defined as follows:

Input: $(\text{Sub}_{j,i}, d_{j,i}, \text{input}_j)$, held by party i of \mathcal{E}_j .

The function F_j :

- (a) If any of the input_j 's are inconsistent, or $\text{Sub}_{j,i} \neq \text{Rec}(\text{crs}_{\text{eq}}^\alpha, c_{j,i}, d_{j,i})$ for any i , where P_α is the i 'th party in committee \mathcal{E}_j (i.e., if any party's share does not agree with the corresponding published commitment), then **abort**.
- (b) Otherwise, let $\text{Sub}_j = \bigoplus_{i=1}^{|\mathcal{E}_j|} \text{Sub}_{j,i}$.
- (c) Evaluate the j 'th module on input_j : that is, $\text{input}_{j+1} := \text{Sub}_j(\text{input}_j)$.

Output: All parties learn input_{j+1} .

At the conclusion of the WLR-MPC execution, each party in \mathcal{E}_j erases all intermediate values generated during the WLR-MPC, keeping only $(\text{Sub}_{j,i}, d_{j,i})$.

Each party in \mathcal{E}_j sends the value of input_{j+1} to all parties in \mathcal{E}_{j+1} (where $\mathcal{E}_{m+1} := \mathcal{P}$ the set of *all* parties).

If any party in \mathcal{E}_{j+1} receives disagreeing values of input_{j+1} from parties in \mathcal{E}_j , then **abort**.

2. Output input_{m+1} as the desired evaluation $f(x)$.

Figure 3: Protocol Π_{Comp} : Compute phase.

Update Phase:

Public inputs: $\mathcal{E}_1, \dots, \mathcal{E}_m$, $\text{crs} = (\{\text{crs}_{\text{eq}}^i, \text{crs}_{\text{nizk}}^i\}_{i=1}^n, \text{crs}_w)$, $\{c_{j,i}\}_{j \in [m], i \in [|\mathcal{E}_j|]}$.

Private inputs: $(\text{Sub}_{j,i}, d_{j,i})$, held by party i of \mathcal{E}_j .

All parties run the Update protocol of the LDS compiler, as follows.

1. Each time the parties in committee \mathcal{E}_j , who are simulating submodule Sub_j , receive a message msg_j from the parties in committee \mathcal{E}_{j-1} , who are simulating submodule Sub_{j-1} , they compute the function G that would have been computed by Sub_j upon receiving the message input_j when running the Update protocol. (The parties in committee \mathcal{E}_1 start with $\text{msg}_1 \triangleq \perp$).

The computation of G is done by running an execution of the λ -weakly leakage-resilient MPC protocol using crs_w to collectively execute the following (randomized) function:

Input: $(\text{Sub}_{j,i}, d_{j,i}, \text{msg}_j)$, held by party i of \mathcal{E}_j ,
 crs , $\{c_{j,i}\}_{i \in [|\mathcal{E}_j|]}$, held by all parties.

The (randomized) function:

- (a) If any of the msg_j 's are inconsistent, or $\text{Sub}_{j,i} \neq \text{Rec}(\text{crs}_{\text{eq}}^\alpha, c_{j,i}, d_{j,i})$ for any i , where P_α is the i 'th party in committee \mathcal{E}_j (i.e., if any party's share does not agree with the corresponding published commitment), then **abort**. Otherwise, let $\text{Sub}_j = \bigoplus_{i=1}^{|\mathcal{E}_j|} \text{Sub}_{j,i}$.
- (b) Evaluate $(\text{Sub}'_j, \text{msg}_{j+1}) \leftarrow G(\text{Sub}_j, \text{msg}_j)$. Here, Sub'_j denotes an updated version of the submodule information, and msg_{j+1} denotes the message to be sent to submodule $j+1$ as dictated by Update.
- (c) Secret share the new value $\text{Sub}'_j = \text{Sub}'_{j,1} \oplus \dots \oplus \text{Sub}'_{j,|\mathcal{E}_j|}$ into $|\mathcal{E}_j|$ shares using the xor secret sharing scheme.
- (d) For each share $\text{Sub}'_{j,i}$ generated in the previous step, compute a new commitment $(c'_{j,i}, d'_{j,i}) \leftarrow \text{Com}(\text{crs}_{\text{eq}}^\alpha, \text{Sub}'_{j,i})$, where P_α is the i 'th party in committee \mathcal{E}_j .

Output: The outputs are as follows.

All parties: $\text{msg}_{j+1}, \{c'_{j,i}\}_{i \in [|\mathcal{E}_j|]}$
 Party i of \mathcal{E}_j : $\text{Sub}'_{j,i}, d'_{j,i}$

At the conclusion of the WLR-MPC execution, each party in \mathcal{E}_j erases all intermediate values generated during the WLR-MPC, keeping only $(\text{Sub}_{j,i}, d_{j,i})$.

2. All the parties of \mathcal{E}_j send msg_{j+1} to *all* parties in \mathcal{E}_{j+1} .
3. Each party in \mathcal{E}_j sends all new commitments $\{c'_{j,i}\}_{i \in [|\mathcal{E}_j|]}$ to every party. If any disagreeing values are sent by parties in \mathcal{E}_j , then **abort**.

At the conclusion of the update phase, each party erases their initial input together with all intermediate values of the update phase.

Figure 4: Protocol Π_{Update} : Update phase.

as shared common knowledge. We denote this ideal functionality by crs .

Ideal functionalities in Hybrid 2: crs.

Hybrid 3. (CRS simulation) The same as Hybrid 2, except that some of the CRS values are generated using the simulation algorithms. More specifically,

- For the first party in each committee, its crs for the equivocal commitment scheme is generated using the simulator; i.e., for each such party P_i , $(\text{crs}_{\text{eq}}^i, \text{trap}^i) \leftarrow \mathcal{S}_{\text{eq}}^{\text{crs}}(1^k)$.
- For each malicious party P_ℓ , we generate its crs for the NIZK proof of knowledge using the simulator, by computing $(\text{crs}_{\text{nizk}}^\ell, \text{trap}^\ell) \leftarrow \mathcal{S}_{\text{nizk}}^{\text{crs}}(1^k)$.
- The crs for the WLR-MPC protocol is simulated by computing $(\text{crs}_w, \text{trap}_w) \leftarrow \mathcal{S}_w^{\text{crs}}(1^k)$.

The remaining crs values are generated honestly, as before. We denote this new ideal functionality by crsSim .

Ideal functionalities in Hybrid 3: crsSim.

Hybrid 4. (MPC security) The same as Hybrid 3, except that the second MPC in the preprocessing phase (which generates a key pair for the FHE scheme, runs the LDS transformation, etc) is replaced by the corresponding ideal (randomized) functionality F_{Pre} . Note that F_{Pre} takes no inputs.

Overall, this hybrid is the same as the real world, except that the preprocessing phase consists only of the execution of Elect and one-time oracle access to crsSim and F_{Pre} .

Ideal functionalities in Hybrid 4: crsSim, F_{Pre} .

Hybrid 5. (WLR-MPC security) The same as Hybrid 4, except each underlying weakly leakage-resilient MPC execution in the decryption cascade is replaced with the ideal functionality F_j that accepts inputs from all parties in \mathcal{E}_j and replies with the evaluation of F_j on these inputs (as described in Figure 3). Similarly, each WLR-MPC execution in the update phase is replaced with the ideal functionality G_j that accepts inputs from parties and replies with the evaluation of G_j on these inputs (as described in Figure 4).

The adversary no longer makes leakage queries of the form $\text{Leak}(i, L)$, as he did in all previous hybrids. Instead, leakage queries are of the form $\text{Leak}(L)$, and are made directly to the ideal functionalities $\{F_j\}, \{G_j\}$. The corresponding ideal functionality evaluates the queried function L on the collection of received inputs from parties. As before, leakage time periods span from the beginning of one Update procedure to the end of the next, and the adversary may make no more than λ leakage queries in any time period.

Ideal functionalities in Hybrid 5: crsSim, $F_{\text{Pre}}, \{F_j\}, \{G_j\}$.

Hybrid 6. (Equivocal commitments) Same as Hybrid 5, except that the ideal functionality F_{Pre} is replaced by a slightly modified functionality F'_{Pre} . Loosely speaking, F'_{Pre} is the same as F_{Pre} , except that for the first party in each committee (which is assumed to be honest), F'_{Pre} generates a *simulated* commitment to the party's secret share.

Explicitly, F'_{Pre} has a trapdoor trap^j , for the first party of each committee \mathcal{E}_j , hardwired into it. Just as F_{Pre} , the functionality F'_{Pre} takes no inputs; it samples a key pair for the FHE scheme, evaluates the LDS transformation of the circuit $\text{Dec}_{\text{sk}}(\cdot)$, and generates secret shares $\text{Sub}_{j,i}$ for each of the resulting secret modules. Further, F'_{Pre} honestly generates a commitment

$$(c_{j,i}, d_{j,i}) \leftarrow \text{Com}(1^k, \text{Sub}_{j,i})$$

to $\text{Sub}_{j,i}$ as usual for the secret share of all *but the first* party in each committee. For the *first* party in each committee (which is assumed to be honest), F'_{Pre} generates a *simulated* commitment

$$(\tilde{c}_{j,1}, \tilde{d}_{j,1}^0, \tilde{d}_{j,1}^1) \leftarrow \mathcal{S}_{\text{eq}}^{\text{com}}(\text{crs}_{\text{eq}}^j, \text{trap}^j),$$

and sets $d_{j,1} = \tilde{d}_{j,1}^{\text{Sub}_{j,1}}$.

Ideal functionalities in Hybrid 6: crsSim, $F'_{\text{Pre}}, \{F_j\}, \{G_j\}$.

Hybrid 7. (Equivocal commitments) Same as Hybrid 6, except that the ideal functionalities $\{G_j\}$ are modified in the same fashion as the step above. Namely, we replace each G_j with a new ideal functionality G'_j with the following differences. G'_j has a trapdoor trap^j , for the first party of each committee \mathcal{E}_j , hardwired into it. G'_j accepts the same inputs as G_j , and carries out the same computation as G_j , with the following exception: For the first party in each committee, instead of honestly generating a commitment to the secret share $\text{Sub}_{j,1}$, the functionality G'_j generates a *simulated* commitment $(\tilde{c}_{j,1}, \tilde{d}_{j,1}^0, \tilde{d}_{j,1}^1) \leftarrow \mathcal{S}_{\text{eq}}^{\text{com}}(\text{crs}_{\text{eq}}^j, \text{trap}^j)$, and sets $d_{j,1} = \tilde{d}_{j,1}^{\text{Sub}_{j,1}}$. (Note that the ideal functionalities $\{F_j\}$ in the decryption cascade do not generate new secret shares and thus do not need to be modified in this fashion).

Ideal functionalities in Hybrid 7: crsSim, $F'_{\text{Pre}}, \{F_j\}, \{G'_j\}$.

Hybrid 8. (Binding of Com) Similar to Hybrid 7, except that all secret shares are eliminated, and committees interact directly with the m modules $(\text{Sub}_1, \dots, \text{Sub}_m)$. More specifically, the following changes are made:

- The ideal functionality crsSim is replaced by a slightly modified functionality crsSim' , which executes exactly as crsSim , but in addition sends to the adversary all trapdoors for simulated equivocal commitment crs values (for the first party in each committee).
- The ideal functionality F'_{Pre} is replaced by a simple ideal functionality F_{pk} that takes no inputs, generates a key pair (pk, sk) for the FHE scheme, and publishes pk .
- The sequence of ideal functionalities $\{F_j\}, \{G'_j\}$, as introduced in the previous steps, are replaced by the corresponding (LDS model) interactions with the m modules $(\text{Sub}_1, \dots, \text{Sub}_m)$ generated by the LDS compiler:

$$(\text{Sub}_1, \dots, \text{Sub}_m) \leftarrow \mathcal{C}(\text{Dec}_{\text{sk}}(\cdot)).$$

Namely,

- The decryption cascade takes place as follows. For each $j = 1, \dots, m$, beginning with \mathcal{E}_1 and

$\text{input}_1 = \hat{y}$, all parties in committee \mathcal{E}_j send input_j to the corresponding module Sub_j . If all input_j 's are consistent, they receive back $\text{input}_{j+1} \leftarrow \text{Sub}_j(\text{input}_j)$. The parties of \mathcal{E}_j then send input_{j+1} to all parties in the next committee \mathcal{E}_{j+1} , who (if the received values are consistent) repeat the same process. At the conclusion of the decryption cascade, the parties of the final committee \mathcal{E}_m send the resulting value input_{m+1} (which is supposedly $f(\vec{x})$) to all parties.

- The update procedure is similar to the decryption cascade. The modules execute the LDS update procedure, interacting with each other via the committees $\mathcal{E}_1, \dots, \mathcal{E}_m$.

Instead of making leakage queries to the ideal functionalities $\{F_j\}, \{G_j\}$, the adversary now makes queries of the form $\text{Leak}(j, L)$, and receives the evaluation of L on the secret state of the j th module, Sub_j . As before, leakage time periods span from the beginning of one **Update** procedure to the end of the next, and the adversary may make no more than λ leakage queries in any time period.

Ideal functionalities in Hybrid 8: $\text{crsSim}', F_{\text{pk}}, \{\text{Sub}_j\}$.

Hybrid 9. (LDS security) Same as Hybrid 8, except that all modules Sub_j are removed. Instead, parties interact with an ideal decryption functionality Dec_{sk} , as described below.

In the preprocessing phase, the parties execute **Elect**, and are given pk and crs values (where some of the crs values are generated with trapdoors, as described in Hybrid 3). The input phase takes place as usual. In the online phase, for each function f that is queried by the adversary, the parties homomorphically compute the corresponding ciphertext $\hat{y} = \text{Eval}_{\text{pk}}(\hat{x}, f)$. All parties in the first committee, \mathcal{E}_1 , send \hat{y} to the ideal decryption functionality $\text{Dec}_{\text{sk}}(\cdot)$ with abort. If all received \hat{y} 's are consistent, the ideal functionality responds by sending the resulting decryption $\text{Dec}_{\text{sk}}(\hat{y})$ to the adversary, where sk is the decryption key that was generated by F_{pk} . If the adversary allows, $\text{Dec}_{\text{sk}}(\hat{y})$ is also sent to all honest parties; otherwise, the experiment concludes in abort. The update phase no longer takes place. No leakage queries are allowed at any point of the experiment.

Ideal functionalities in Hybrid 9: $\text{crsSim}', F_{\text{pk}}, \text{Dec}_{\text{sk}}$.

Hybrid 10. (Soundness/PoK of NIZK, certifiability of FHE) Differs from Hybrid 9 in the following ways:

- The ideal functionalities $\text{crsSim}', F_{\text{pre}}$, and the execution of **Elect**, are removed from the preprocessing phase.
- The input phase no longer takes place.
- The ideal decryption functionality Dec_{sk} is replaced by the ideal-world functionality **Evaluate**, which takes input x_i from each party and evaluates functions f queried by the adversary on the set of all parties' inputs \vec{x} , as defined in Section 3.1.

- In addition, the adversary is given as auxiliary input

$$z' := (\text{pk}, \{\hat{x}_i, \text{crs}_{\text{nizk}}^i, \pi_i\}_{i \notin M}),$$

where $(\text{pk}, \text{sk}) \leftarrow \text{Gen}(1^k)$, and for each honest party P_i , the triple $(\text{crs}_{\text{nizk}}^i, \hat{x}_i, \pi_i)$ is computed using the *real* input x_i of P_i . That is, the values in the triple are computed by $\text{crs}_{\text{nizk}}^i \leftarrow \text{Gen}_{\text{nizk}}(1^k)$; $r_i \leftarrow R$; $\hat{x}_i = \text{Enc}_{\text{pk}}(x_i; r_i)$; $\pi_i \leftarrow P(\text{crs}_{\text{nizk}}^i, (\hat{x}_i, \text{pk})(x_i, r_i))$.

Overall, Hybrid 10 is the following.

Parties begin by submitting their inputs to the ideal functionality **Evaluate**. More specifically, each *honest* party P_i submits his input x_i . The adversary is given the corresponding auxiliary input z' , computed as a function of the honest parties' inputs $\{x_i\}_{i \notin M}$. Upon receiving z' , the adversary submits the inputs of *malicious* parties to **Evaluate**.

The preprocessing and input phases no longer take place. During the online phase, for each function f that is queried by the adversary, **Evaluate** responds by sending the adversary the evaluation of f on the set of all submitted inputs (x_1, \dots, x_n) . If the adversary allows, the evaluation is also sent to all honest parties; otherwise, the experiment concludes in abort.

Note that Hybrid 10 is *nearly* the ideal-world experiment. Indeed, the only difference is that the adversary is given the auxiliary input z' .

Ideal functionalities in Hybrid 10: **Evaluate**.

Hybrid 11. (Security of FHE, ZK of NIZK) The ideal world: i.e., the adversary *only* receives $f(x_1, \dots, x_n)$ for each f selected to be computed. Note that this is the same as Hybrid 10, except that the adversary no longer receives the auxiliary input. (See Section 3.1 for the detailed experiment).

The output of each hybrid experiment consists of the outputs of all parties, where honest parties output in accordance with the dictated protocol, and malicious parties may output any efficiently computable function of the view of the adversary. For every adversary \mathcal{A}_ℓ with auxiliary input $z \in \{0, 1\}^*$ running in hybrid experiment ℓ with initial inputs \vec{x} , we denote the output of the corresponding hybrid ℓ experiment by

$$\text{HYB}_\ell(\mathcal{A}_\ell, 1^k, z, \{x_i\}_{i=1}^n).$$

It remains to prove that for every $\ell = 0, \dots, 10$ and for every adversary \mathcal{A}_ℓ running in Hybrid ℓ , there exists an adversary $\mathcal{A}_{\ell+1}$ running in Hybrid $(\ell + 1)$ such that

$$\text{HYB}_\ell(\mathcal{A}_\ell, 1^k, z, \{x_i\}_{i=1}^n) \approx_c \text{HYB}_{\ell+1}(\mathcal{A}_{\ell+1}, 1^k, z, \{x_i\}_{i=1}^n).$$

Note that once we show this, the theorem will follow, as this will imply that for each adversary \mathcal{A} in the real-world experiment (Hybrid 0), there is an adversary \mathcal{A}_{11} in the ideal-world experiment (Hybrid 11), such that

$$\text{HYB}_0(\mathcal{A}, 1^k, z, \{x_i\}_{i=1}^n) \approx_c \text{HYB}_{11}(\mathcal{A}_{11}, 1^k, z, \{x_i\}_{i=1}^n)$$

as desired. We defer these indistinguishability proofs to the full version of this manuscript.

□

6. REFERENCES

- [ADN⁺10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.
- [ADW09] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.
- [AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.
- [AK96] Ross Anderson and Markus Kuhn. Tamper resistance: a cautionary note. In *WOEC’96: Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce*, pages 1–11, 1996.
- [BCG⁺11] Nir Bitansky, Ran Canetti, Shafi Goldwasser, Shai Halevi, Yael Tauman Kalai, and Guy N. Rothblum. Program obfuscation with leaky hardware. In *ASIACRYPT*, 2011.
- [BCH11] Nir Bitansky, Ran Canetti, and Shai Halevi. Leakage tolerant interactive protocols. Cryptology ePrint Archive, Report 2011/204, 2011.
- [BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications (extended abstract). In *STOC*, pages 103–112, 1988.
- [BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO*, pages 1–20, 2010.
- [BGG⁺11] Elette Boyle, Sanjam Garg, Shafi Goldwasser, Abhishek Jain, Yael Tauman Kalai, and Amit Sahai. Leakage-resilient multiparty computation. Manuscript, 2011.
- [BGK11] Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. In *DISC*, 2011.
- [BGV11] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. Fully homomorphic encryption without bootstrapping. ECCC, Report 2011/111, 2011.
- [BKKV10] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.
- [BSMP91] Manuel Blum, Alfredo De Santis, Silvio Micali, and Giuseppe Persiano. Noninteractive zero-knowledge. *SIAM J. Comput.*, 20(6):1084–1118, 1991.
- [BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In *EUROCRYPT*, 2011.
- [BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) lwe. In *FOCS*, 2011.
- [CIO98] Giovanni Di Crescenzo, Yuval Ishai, and Rafail Ostrovsky. Non-interactive and non-malleable commitment. In *STOC*, pages 141–150, 1998.
- [CLOS02] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *STOC*, pages 494–503, 2002.
- [DGK⁺10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In *TCC*, pages 361–381, 2010.
- [DHLW10a] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.
- [DHLW10b] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.
- [DHP11] Ivan Damgård, Carmit Hazay, and Arpita Patra. Leakage resilient two-party computation. Cryptology ePrint Archive, Report 2011/256, 2011.
- [DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In *STOC*, pages 621–630, 2009.
- [DLWW11] Yevgeniy Dodis, Allison Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. In *FOCS*, 2011.
- [DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.
- [DP10] Yevgeniy Dodis and Krzysztof Pietrzak. Leakage-resilient pseudorandom functions and side-channel attacks on feistel networks. In *CRYPTO*, pages 21–40, 2010.
- [DS83] Danny Dolev and H. Raymond Strong. Authenticated algorithms for byzantine agreement. *SIAM J. Comput.*, 12(4):656–666, 1983.
- [Fei99] Uriel Feige. Noncryptographic selection protocols. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, 1999.
- [FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.
- [FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317, 1990.
- [FRR⁺10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the

- computationally-bounded and noisy cases. In *EUROCRYPT*, pages 135–156, 2010.
- [FS89] Uriel Feige and Adi Shamir. Zero knowledge proofs of knowledge in two rounds. In *CRYPTO*, pages 526–544, 1989.
- [Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.
- [GJS11] Sanjam Garg, Abhishek Jain, and Amit Sahai. Leakage-resilient zero knowledge. In *CRYPTO*, pages 297–315, 2011.
- [GMO01] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In *CHES*, pages 251–261, 2001.
- [GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.
- [GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Perfect non-interactive zero knowledge for np. In *EUROCRYPT*, pages 339–358, 2006.
- [GR10] Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In *CRYPTO*, pages 59–79, 2010.
- [GR12] Shafi Goldwasser and Guy N. Rothblum. How to compute in the presence of leakage. *Electronic Colloquium on Computational Complexity (ECCC)*, 19, 2012.
- [HSH⁺08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, pages 45–60, 2008.
- [ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.
- [JV10] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In *CRYPTO*, pages 41–58, 2010.
- [Koc96] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In *CRYPTO*, pages 104–113, 1996.
- [KP10] Eike Kiltz and Krzysztof Pietrzak. Leakage resilient elgamal encryption. In *ASIACRYPT*, pages 595–612, 2010.
- [KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.
- [LLW11] Allison Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *STOC*, 2011.
- [LRW11] Allison Lewko, Yannis Rouselakis, and Brent Waters. Achieving leakage resilience through dual system encryption. In *TCC*, 2011.
- [MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.
- [MTVY11] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In *EUROCRYPT*, 2011.
- [NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.
- [OST06] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache attacks and countermeasures: The case of aes. In *CT-RSA*, pages 1–20, 2006.
- [Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In *EUROCRYPT*, pages 462–482, 2009.
- [QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In *E-smart*, pages 200–210, 2001.
- [Yao82] Andrew C. Yao. Theory and applications of trapdoor functions. In *Proc. 23rd FOCS*, pages 80–91, 1982.

Glossary of Terms

ABE	Attribute Based Encryption Scheme
BDD	Bounded Distance Decoding Assumption
CPRF	Constrained Pseudorandom Functions
CRS	Common Reference String
CVP	Closest Vector Problem
DDH	Decisional Diffie-Hellman
FE	Functional Encryption Scheme
FHE	Fully Homomorphic Encryption
F-PRFs	Functional Pseudorandom Functions
GCD	Greatest Common Divisors
HE	Homomorphic Encryption
HELib	Homomorphic Encryption Library
IO	Indistinguishable Obfuscation
Leveled FHE	FHE Evaluation of Circuits of a-priori Bounded Depth
LWE	Learning with Errors
MPC	Multi Party Computation
Multikey FHE	Fully Homomorphic Encryption utilizing multiple, unrelated keys
NTRU	N-th Order Truncated Ring Encryption Scheme
PE	Predicate Encryption Scheme
PI	Principle Investigator
PRFs	Pseudorandom Functions
PROCEED	Programming Computation on Encrypted Data
Ring LWE	Ring Learning with Errors
RLWE	Ring Learning with Errors
SHE	Somewhat Homomorphic Encryption
SIMD	Single Instruction / Multiple Data
SIS	Short Integer Solution
SKI	Symmetric Key Infrastructure
SSS	Sparse Subset Sum Assumption
SVP	Shortest Vector Problem